THE APPLICATIONS OF DISCRETE OPTIMAL TRANSPORT IN PATH PLANNING AND DATA CLUSTERING

A Dissertation
Presented to
The Academic Faculty

By

Haoyan Zhai

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
School of School of Mathematics

Georgia Institute of Technology

August 2019

THE APPLICATIONS OF DISCRETE OPTIMAL TRANSPORT IN PATH PLANNING AND DATA CLUSTERING

Approved by:

Prof. Luca Dieci
School of Mathematics
Georgia Institute of Technology

Prof. Sung Ha Kang
School of Mathematics
Georgia Institute of Technology

Prof. Martin Short
School of Mathematics
Georgia Institute of Technology

Prof. Fumin Zhang
School of Electrical and Computer Engineering
Georgia Institute of Technology

Prof. Haomin Zhou, Advisor
School of Mathematics
Georgia Institute of Technology

Date Approved: April 17, 2019

Where there is a will, there is a way.

To

My Parents and wife *Li*.

ACKNOWLEDGEMENTS

It has been five years since I arrived Georgia Tech. During my Ph.D life, I visited the beautiful world of the applied mathematics and I would not achieve what I have and finish this thesis without the help from people around me.

First of all, I would like to express my deepest gratitude to my advisor, Professor Haomin Zhou, for all his guides and supports of the researches and papers. I really enjoy the extensive rewarding discussions with him as well as his incisive and novel understanding of the problems. His meticulous care make my five-year Ph.D career productive, rich and colorful. At the same time, I would convey my gratefulness to him for being my life mentor. I own him beneficial advice and understanding on my future career and life. I cannot sufficiently voice my appreciation to him and I feel lucky to have him as my advisor.

Special thanks goes to Professor Fumin Zhang and Magnus Egerstedt. The collaboration not only provides me opportunities to enrich my knowledge of the world of applications, but also exhibits new directions for my researches. Many thanks to their constructive advice on the researches and affirmative encouragements. Without their help, I would not have the chance to practically extend mathematical theories a real life problem and see the exciting results in my research.

I am extremely grateful to Professors Luca Dieci, Sung Ha Kang and Martin Short for sparing time to serve as my committee members. I appreciate their valuable and insightful questions and suggestions.

I would like to thank Dr. Wuchen Li and my collaborator, Mengxue Hou, for the continuous guides and discussions. They offer me professional advice and thoughts which tremendously push forward the progress of the projects. Their comments always allow me to see the problems in different aspects and inspire me to come up with new ideas. I have a good time working with them and would like to keep the collaboration in the future.

Meanwhile, I want to offer my thanks to my graduate fellows and friends. Allow me to

specially thank Jiangning Chen, Qianli Hu, Xin Xing, Jiaqi Yang, Shu Liu, Ruilin Li, Yaohua Zang, Shaojun Ma, Jun Lu and Fan Zhou for years of friendship and many interesting discussions in both mathematics and life.

Last but not least, I convey my deep thanks to my parents and my wife Li, without the constant supporting and caring of whom, I would not have the opportunity to have gone so far.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SUMMARY

Optimal transport introduces the concept of Wasserstein distance, which has been widely used in various applications in computational mathematics, machine learning as well as many areas in engineering. Meanwhile, control theory and path planning is an active branch in mathematics and robotics, focusing on algorithms that calculate feasible or optimal paths for robotic systems. In this thesis, we use the properties of the gradient flows induced by Wasserstein-2 metric to design algorithms to handle different types of path planning and control problems. Also, we define the Wasserstein K-means problems on graphs and propose an efficient algorithm to solve it.

In Chapter 2, we provide necessary mathematical concepts and results that form the basis of this thesis. It contains brief surveys of optimal controls, gradient flows, intermittent diffusion, Fokker-Planck equations and optimal transport on both continuous and discrete spaces.

Chapter 3 gives an algorithm to handle the path planning problem in unknown environments. We develop a deterministic approach with finite-step convergence guarantee. Also, there is a theoretical relation between this algorithm and the Fokker-Planck equations, which bounds the searching region of the algorithm. We use numerical examples to show the efficiency of the algorithm as well as to support the theoretical results.

In Chapter 4, we generalize the algorithm to solve the general control problem in the unknown environments and similar convergence results can be proven. Besides, there is an evidence that the algorithm is guided by the evolution of Fokker-Planck equation, and we use experiments to demonstrate our theorems.

In Chapter 5, we study the optimal path planning in flow field. In this case, the objective function, the traveling time or kinetic energy, is to be minimized with a given flow field. Following the idea of method of evolving junctions, we first transform the original infinite dimensional optimal control into a finite dimensional global optimization problem by in-

troducing junctions located only on the discontinuity positions of the dynamics. To handle the global optimization, intermittent diffusion is used here to guarantee the completeness of the method.

At last, in Chapter 6 we define the discrete Wasserstein-(1,p) distance that depends on the graph structure. With this distance function, we further propose the Wasserstein K-means problem on a general graph and provide an algorithm in the framework of Lloyd method. The key part of the algorithm is the calculation of discrete Wasserstein-(1,p) distance and the gradient flow induced by Wasserstein-2 metric to solve an optimization with objective function being a linear combination of Wasserstein-(1.p) distance. Examples and simulation results are provided in the last part of this chapter.

CHAPTER 1

INTRODUCTION

Wasserstein distance in optimal transport theory provides an approach to equip the probability manifold $\mathcal{P}(\mathbb{R}^d)$ with a metric, which induces a gradient operator. Thus, a gradient flow can be defined on $\mathcal{P}(\mathbb{R}^d)$ for arbitrary functional $\mathcal{F}(\rho) : \mathcal{P}(\mathbb{R}^d) \to \mathbb{R}$ as below

$$\rho_t = -grad\mathcal{F} = div\left(\rho\nabla\frac{\delta\mathcal{F}}{\delta\rho}\right). \tag{1.1}$$

For example, if one defines $\mathcal{F}$ to be the free energy

$$\mathcal{F}(\rho) = \int_{\mathbb{R}^d} \rho(x)V(x) + \beta\rho(x)\log\rho(x)dx,$$

the gradient flow is the famous Fokker-Planck equation given by:

$$\begin{aligned}\rho_t(x,t) &= div\left(\rho(x,t)\nabla(V(x) + \beta\log\rho(x,t))\right)\\ &= div(\rho(x,t)\nabla V(x)) + \beta\Delta\rho(x,t).\end{aligned}$$

Using the properties of the gradient flows in optimal transport, we can design algorithms to handle various types of problems. In this thesis, we exhibit algorithms to solve certain kinds of control problems as well as define and solve the well-known K-means problem on finite simple graphs.

1.1 Dynamical Path Planning Method For Control Problems With Partially Given Constraints

A control system $(\mathcal{T}, \Omega, \mathcal{U}, \kappa)$ contains four major components: $\mathcal{T}$ is a time set, which is a subgroup of $\mathbb{R}_+$; Ω is a state space, which can be a subset of $\mathbb{R}^d$, or a Riemannian manifold;

$\mathcal{U}$ is the control space; and $\kappa : \mathcal{D}_\kappa \to \Omega$ is the transition map defined on

$$\mathcal{D}_\kappa = \left\{(\tau, \sigma, x, u) : \tau, \sigma \in \mathcal{T}, x \in \Omega, u \in \mathcal{U}^{[\sigma,\tau)}\right\}.$$

Control theory aims to study the properties of the transition map κ and design the control function u to influence the trajectory of certain objects to achieve desired goal, that is, in an interval $[\sigma, \tau)$, we want to find $u \in \mathcal{U}^{[\sigma,\tau)}$, such that $\kappa(\sigma, \tau, x_0, u)$ meets specific conditions with initial state x_0 given. In many continuous circumstances, κ can be described by a system of ordinary differential equations. If we fix x_0 at time σ and define $\gamma(t) = \kappa(\sigma, t, x_0, u)$, there exists a (Lipschitz) function f such that $\dot{\gamma} = f(x, u, t)$. Meanwhile, sometimes additional constraints are added to the system to regularize the state and control variables. In this thesis, we only consider the holonomic constraints, which can be represented as $\phi(x) \geq 0$ for $x \in \Omega$. Given certain initial x_0 and target x_f states, path planning focuses on planning a feasible or an optimal trajectory γ such that $\gamma(0) = x_0$, $\gamma(\tau) = x_f$ with $\gamma(t) = \kappa(0, t, x_0, u)$ for some control function u.

In the control system, we set $\mathcal{T} = \mathbb{R}_+$ and let the transition map κ determined by the following ordinary differential equation:

$$\frac{d}{d\tau}\kappa(\tau, \sigma, x_0, u) = f(x, u),$$

where we assume f is a Lipschitz function. Then we can rewrite the system into the following time-invariant continuous one $(\Omega, \mathcal{U}, f)$, providing that Ω is the state space equipped

with a distance function $d(\cdot,\cdot)$, with holonomic constraints ϕ. Thus, we have

$$\dot{x} = f(x,u)$$
$$\phi(x(t)) \geq 0$$
$$x(t) \in \Omega, u(t) \in \mathcal{U}.$$

Further, we assume there is another set of constraints $\hat{\psi}(x)$ satisfying

$$\hat{\psi}(x,\gamma,t) = \begin{cases} \psi(x) & \text{if } d(x,\gamma,\tau) \leq R \text{ for some } \tau \leq t \\ 0 & \text{otherwise} \end{cases},$$

where $\psi(x) \leq 0$ is another set of constraints with R given priorly. In this thesis, we study the problem described above and propose a graph-based, potential guided algorithm to search for a feasible path combined with a corresponding set of control variables. The algorithm has three main parts: Graph Generation, Path Finding and Trajectory Extension (or Environment Updating). Among them, Graph Generation is the core part, which is potentially guided by the evolution of the Fokker-Planck equation (FPE) in the Optimal Transport Theory

$$\rho_t(x,t) = div(\rho(x,t)\nabla\psi(x)) + \beta\Delta\rho(x,t).$$

There are many ways to formulate the optimal transport problem such as using linear programming or partial differential equations, among which Benamou-Brenier Formulation rewrites the mass transport problem to be a optimal control problem and defines a Riemannian metric with Wasserstein-2 distance

$$W_2(\rho^0,\rho^1) = \inf_v \left\{ \left(\int_\Omega \int_0^1 \|v\|^2 \rho(x,t) dt dx \right)^{\frac{1}{2}} : \rho_t + div(\rho v) = 0, \rho(0) = \rho^0, \rho(1) = \rho^1 \right\},$$

where $\|\cdot\|$ is the standard Euclidean metric. Taking advantage of this metric, FPE can

be rewritten as a gradient flow of the free energy and can reach the Gibbs' distribution asymptotically. The design of our algorithm is based on this property, and we show that it is a deterministic graph based procedure with guaranteed convergence and returns a feasible path if there exists one.

In Chapter 3, we discuss this problem in a simple Euclidean space to show how this algorithm works and some of its properties. Later in Chapter 4, we generalize the algorithm in a control setup, where special cases are given to present detailed strategies to handle the problem. Meanwhile, two different points of views are provided in these two chapters respectively to show the deep relation between our algorithm and the evolution of Fokker-Planck equation.

1.2 Optimal path planning in partitioned flow fields

In the Fokker-Planck equation (1.1), if we treat ρ as the density function, we can have the corresponding stochastic differential equation in the particle level:

$$dx_t = -\nabla V(x_t)dt + \sqrt{2\beta}dW_t. \tag{1.2}$$

As the Fokker-Planck is the gradient flow heading to the Gibbs' distribution, (1.2) shares a similar property, that is the particle has the largest probability to reach the global minimizer of the potential function $V(x)$. Based on this, a global minimization problem can be solved by the intermittent diffusion method [1]. In Chapter 5, we will discuss our approach to the infinite dimensional optimal control problem with no additional state constraints being posed. Meanwhile, we specify the dynamics as $f(x, u) = u + v$ where v is a piece-wise

constant vector. Formally, the optimal path planning problem is as below

$$\begin{aligned} &\min_{v,T} \int_0^T L(x,v)dt \qquad (1.3)\\ s.t.\quad &\dot{x} = v + u,\\ &x(0) = x_0,\\ &x(T) = x_f,\\ &\max_{t\in[0,T]} \|v\| \leq V.. \end{aligned}$$

We let $L(x,u) = 1$ or $L(x,u) = \|u\|^2 + C$, which corresponds to the minimal time and minimal energy respectively. Following the idea of the Method of Evolving Junctions [2], we prove the optimal structure of sub-problems within each constant flow region can be explicitly expressed. The optimal solution can be determined simply by finite number of points on the boundary of partitioned regions, which are separated according to the discontinuous positions of v. Those points are called junctions, with which, the original problem is reduced to a finite dimensional optimization. To calculate its minimizer, we use intermittent diffusion method. In Chapter 5, we derive the problem in detail from a practical glider optimal path planning problem, give corresponding theoretical results and provide different types of examples to show the performance of our algorithm.

1.3 Wasserstein K-means on graph via optimal transport

Recall that the intermittent diffusion methods in Section 1.2 can be treated as an evolution of a stochastic differential equation, and the FPE in Section 1.1 can be further written as a gradient flow format (1.1). If we let $\mathcal{F}(\rho)$ be a general functional defined on the probability manifold, (1.1) becomes the Wasserstein gradient flow for arbitrary functionals. In this thesis, we consider linear combinations of the orientation-based discrete Wasserstein-$(1,p)$

distance

$$W_{1,p}(\rho^0, \rho^1) = \inf_u \left\{ \|u\|_{1,p} : div(u) + \rho^1 - \rho^0 = 0 \right\},$$

defined on a connected simple graph $G = (V, E)$. With this, we propose the K-means problem on the discrete probability space, aiming to cluster different distributions supported on G to a fixed number of classes. To solve this problem, we follow the famous Llyod algorithm to design our method, which iterates over two major steps: The first step is to calculate the discrete Wasserstein-$(1, p)$ distance between centroids and data points, then assign each data point to the class with shortest distance. The second step is to re-assign the location of centroid within each class using all data points belonging to the class. The second step can be written in an optimization problem with objective function a linear combination of Wasserstein-$(1.p)$ distance. To solve this, we use the gradient flow induced by Wasserstein-2 metric on graphs. We will give the formulations, algorithms and experiment results in Chapter 6.

CHAPTER 2

PRELIMINARY IN MATHEMATICS

In this chapter, we introduce the basic mathematics needed in this thesis, including optimal control, intermittent diffusion, gradient flows, Fokker-Planck equations, and optimal transport. These are highly related topics and form the base of our algorithms. And we ignore the regularity issue during the calculation to give straight forward explanations.

2.1 Optimal control

Optimal control is an infinite dimensional optimization problem, aiming to find a control law for a given dynamical system to optimize a functional with certain constraints. In this thesis, we introduce two main techniques to handle a class of problems and the formulation is given as below:

$$\inf_{u,T} \left\{ \int_0^T L(x,u)dt : \dot{x} = f(x,u), x(0) = x_0, x(T) = x_1 \right\}, \tag{2.1}$$

where x_0, x_1 are given states, the running cost $L : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ is the Lagrangian and the dynamics f is a Lipschitz continuous function. Also, we denote the cost functional as

$$J(u,T) = \int_0^T L(x,u)dt.$$

If the dynamics $f(x,u) = u$, meaning that $\dot{x} = u$, and we fix $T = 1$, we can use the calculus of variation to obtain a set of ordinary differential equations (ODE's) for a local minimizer of $J(x)$. We denote $(x(t), u(t))$ a local minimizer and $h(t)$ an arbitrary function with $h(0) = h(1) = 0$. With $x_s(t) = x(t) + sh(t)$, we have $J(x_s)$ as a function of x with

$s = 0$ being a critical point, leading to

$$\begin{aligned} 0 &= \frac{d}{ds} J(x_s)|_{s=0} \\ &= \int_0^1 \left(\nabla_x L(x,u) h(t) + \nabla_u L(x,u) \dot{h}(t) \right) dt \\ &= \int_0^1 \left(\nabla_x L(x,u) - \frac{d}{dt} \nabla_u L(x,u) \right) h(t) dt. \end{aligned}$$

Since h is arbitrarily chosen, the minimizer satisfies

$$\nabla_x L(x,u) - \frac{d}{dt} \nabla_u L(x,u) = 0, \tag{2.2}$$

which is called the Euler-Lagrange equation. The left-hand-side of (2.2) is the first variation formula of J with respect to x, which is denoted as $\frac{\delta}{\delta x} J$ in this thesis.

Moreover, the Euler-Lagrange equation can be written into a Hamiltonian system format. To do so, we construct the Hamiltonian H by the Legendre transform:

$$\begin{aligned} p &= \nabla_u L(x,u), \\ \mathcal{H}(x,p,u) &= p \cdot u - L(x,u), \\ H(p,x) &= \sup_{u \in \mathbb{R}^d} \mathcal{H}(x,p,u) \end{aligned}$$

through which, (2.2) becomes

$$\begin{aligned} \dot{p}(t) &= -\nabla_x H(t, p(t), x(t)) \\ \dot{x}(t) &= \nabla_p H(t, p(t), x(t)). \end{aligned} \tag{2.3}$$

In general to solve (2.1), we have the local minimizer (x, p, u) satisfying

$$\begin{aligned}
&\dot{x}(t) = \nabla_p \mathcal{H}(x(t), p(t), u(t)) \\
&\dot{p}(t) = -\nabla_x \mathcal{H}(x(t), p(t), u(t)) \\
&\mathcal{H}(x(t), p(t), u(t)) = \max_{a \in \mathbb{R}^d} \mathcal{H}(x(t), p(t), a) \quad (0 \le t \le \tau) \\
&\mathcal{H}(x(t), p(t), u(t)) = 0 \quad (0 \le t \le \tau)
\end{aligned}$$

where τ is the first time the trajectory $x(\cdot)$ hits the target state x_1. And this method is called the Pontryagin maximum principle.

The other method is dynamic programming, which involves the calculation of the Hamilton-Jacobi-Bellman equation, a non-linear partial differential equation (PDE). This method gives us a feedback law for the control variable u and provides the global minimizer of (2.1). Starting at the state x_0, we define a cost-to-go function

$$\Phi(t, x) = \inf_u \left\{ \int_0^t L(x(s), u(s))dt : x(0) = x_0, x(t) = x \right\}.$$

By Bellman's principle of optimality, the partial trajectory of optimal solution is also optimal. Going from t to $t + h$, we have

$$\Phi(t + h, x(t + h)) \le \Phi(t, x(t)) + \int_t^{t+h} L(x(s), v(s))ds,$$

where $\dot{x}(s) = f(x(s), v(s))$ for $x \in [t, t + h]$ and v be an arbitrary control variable. Now we rearrange the above equation and divide by $h > 0$:

$$\frac{\Phi(t + h, x(t + h)) - \Phi(t, x(t))}{h} - \frac{1}{h} \int_t^{t+h} L(x(s), v(s))ds \le 0.$$

By letting $h \to 0$,

$$\Phi_t(t,x) + \nabla_x \Phi(t,x)\dot{x} - L(x,v) \leq 0$$
$$\Longrightarrow \Phi_t(t,x) + \nabla_x \Phi(t,x) \cdot f(x,v) - L(x,v) \leq 0.$$

To prove 0 can be reached, we use the optimal solution (x,u) on $[t,t+h]$ to have the cost from t to $t+h$ as

$$\int_t^{t+h} L(x(s),u(s))ds$$

and by Bellman's principle,

$$\Phi(t+h,x(t+h)) = \Phi(t,x) + \int_t^{t+h} L(x(s),u(s))ds.$$

Via conducting the same calculation, we have

$$\Phi_t(t,x) + \nabla_x \Phi(t,x) \cdot f(x,u) - L(x,u) = 0.$$

From here, we can write the Hamilton-Jacobi-Bellman equation to be

$$\Phi_t(t,x) + \max_{u \in \mathbb{R}^d} (\nabla_x \Phi(t,x) \cdot f(x,u) - L(x,u)) = 0$$

If it is the case that $L(x,u) = \|u\|^2$ with $f(x,u) = u$, the Hamilton-Jacobi-Bellman equation is simplified as

$$\Phi_t(t,x) + \frac{1}{2}\|\nabla_x \Phi(t,x)\|^2 = 0$$

with $\dot{x} = \nabla_x \Phi(t,x)$.

2.2 Gradient flows

Gradient flows are evolutionary systems related with a function $V(x)$ defined in a space. Written in the form of

$$dx_t = -\nabla V(x_t)dt, \tag{2.4}$$

the gradient flow admits a Lyapunov function $V(x)$ since

$$\frac{d}{dt}V(x) = -(\nabla V(x) \cdot \nabla V(x)) \leq 0.$$

Thus, if $V(x)$ is strictly convex, (2.4) converges to the global minimizer of $V(x)$, which provides a natural way to solve the optimization problem

$$\min_{x \in \mathbb{R}^d} V(x).$$

In numerical methods, the steepest descent method follows the same philosophy.

Moreover, if the space is a general Riemannian manifold $(\mathcal{M}, g)$ where g defines an inner product on the tangent bundle $T\mathcal{M}$ with $g_x(\cdot, \cdot) : T_x\mathcal{M} \times T_x\mathcal{M} \to \mathbb{R}$, the differential structure induces the gradient operator on $\mathcal{M}$. For an arbitrary energy function $V : \mathcal{M} \to \mathbb{R}$, $gradV(x) \in T_x\mathcal{M}$ is defined as

$$g(gradV, u) = dV(u)$$

for $u \in T\mathcal{M}$ and dV the differential of V. Here, $dV(u)$ means that dV operates on u. Therefore, the gradient flow of V on $(\mathcal{M}, g)$ is

$$dx_t = -gradV(x_t)dt.$$

2.3 Intermittent Diffusion

In many cases, $V(x)$ is not convex, and gradient flows only give us local minimizers. To overcome this problem, one can use the intermittent diffusion (ID) method [1], which proposes to solve the minimization by running the stochastic differential equation

$$dx_t(\omega) = -\nabla V(x_t(\omega))dt + \sigma(x,t)dW_t \tag{2.5}$$

where $t \in [0,\infty]$, W_t is the standard Brownian motion in $\mathbb{R}^d$, ω is a random event in the Wiener space of W_t, and $\sigma(x,t)$ is chosen to be a piece-wise constant function

$$\sigma(x,t) = \sum_{j=1}^{N} \sigma_j \chi_{[S_j,T_j]}(t),$$

where $0 = S_1 < T_1 < S_2 < T_2 < \cdots < \cdots < S_N < T_N < S_{N+i} = T$ and $\chi_{[S_j,T_j]}$ is the characteristic function on the interval $[S_j,T_j]$. The idea behind is that when $\sigma = 0$, (2.5) becomes simply a gradient flow and converges to a local minimizer. When $\sigma > 0$, the Brownian motion controls the state variable $x(\omega)$ to jump out of the local trap. The algorithm can be stated as Algorithm 1.

This algorithm has the following convergence property and we will skip the proof that can be found in [1, 3]:

Theorem 2.3.1. *Let Q be the set of global minimizers, U a small neighborhood of Q and X^{opt} the optimal solution obtained by ID. Then for any given $\epsilon > 0$, there exists $\tau > 0, \sigma_0 > 0, N_0 > 0$ such that if $T_i - S_i > \tau$, $\sigma_i < \sigma$, $i = 1, \cdots, N$ and $N > N_0$,*

$$\mathbb{P}(X^{opt} \in U) \geq 1 - \epsilon$$

Algorithm 1: Intermittent Diffusion

Data: The function f to optimize; number of intermittent diffusion intervals N; maximal diffusion time T_{max} and maximal diffusion strength σ_{max}

Randomly generate an initial state x^0 and let $x^{opt} = x^0$;

for $i = 1, 2, \cdots, N$ **do**

  Uniformly sample $\sigma \in [0, \sigma_{max}]$ and $T \in [0, T_{max}]$;

  Compute the stochastic differential equation for $t \in [0, T]$

$$dx_t(\omega) = -\nabla V(x_t(\omega))dt + \sigma dW_t, \;\; x(0) = x_{opt}$$

  and record the final state $x^T = x_T(\omega)$;

  Compute the gradient flow until converge:

$$\dot{x}_t = -\nabla V(x_t), \;\; x_0 = x^T$$

  and record the final state x^i. If $V(x^i) < V(x^{opt})$, $x^{opt} = x^i$;

end

return $x^1, \cdots, x^N$ *as* N *local minimizers and* x^{opt} *the global minimizer;*

2.4 Fokker-Planck equations

If we further generalize (2.5) into the form of

$$dX_t = f(X_t)dt + \sqrt{2\beta}A(X_t)dW_t, \;\; X_t \in \mathbb{R}^d,$$

we can also describe the evolution of the density function. Here, we define the transition probability of the Markov process X_t as

$$\rho(t, x)dx = \mathbb{P}(X_t \in dx | X_0)$$

where $x \in \mathbb{R}^d$ and $t > 0$, then the corresponding Fokker-Planck equation is

$$\frac{\partial \rho}{\partial t} + div(f(x)\rho) = \beta div(AA^T \nabla \rho), \;\; x \in \mathbb{R}^d,$$

where $AA^T = A(x)A(x)^T$ is a non-negative definite diffusion matrix and $f(x) \in \mathbb{R}^d$ is the drift vector function. Along the evolution of the Fokker-Planck equation, the density

function $\rho(x,t)$ will keep non-negative and conserve the total probability.

Specifically, if $A(x) = 0$ and $f(x) = -\nabla V(x)$, the Fokker-Planck equation admits an equilibrium, named Gibbs measure, which has an explicit formula

$$\rho^* = \frac{1}{K} e^{-\frac{V(x)}{\beta}}, \quad K = \int_{\mathbb{R}^d} e^{-\frac{V(x)}{\beta}} dx.$$

When β approaches 0, ρ^* tends to a Dirac mass, concentrated on the global minimizers of the potential function. Based on this property, many global optimization techniques are designed, including simulated annealing, intermittent diffusion and so on.

Besides, instead of $\mathbb{R}^d$, the state space has various of choices such as a bounded open set with either zero-flux or periodic boundary conditions, and can also be a Riemannian manifold, where a differential structure is provided.

2.5 Optimal transport

The optimal transport theory is an active branch of modern mathematics studying how to transport one probability distribution to another with the optimal cost. The original optimal transport problem was proposed by Monge as following:

$$C(\mu,\nu) = \inf_T \int_X c(x, T(x)) d\mu(x),$$

where X is a d-dimensional Polish space and the minimization is taken over the set of all measurable maps T such that $T\#\mu = \nu$. Here $T\#\mu$ is the push forward measure defined as

$$T\#\mu(A) = \mu(T^{-1}(A))$$

for all measurable set $A \subset X$ and $c(x,y)$ is the cost function for transporting one unit mass from x to y with μ, ν being two probability measures supported on continuous states. The states can be any metric space, Riemannian manifold etc. Later a relaxation version of the

problem is given as

$$C(\mu,\nu) = \inf_{\pi\in\Pi(\mu,\nu)} \iint_{X\times Y} c(x,y)d\pi(x,y), \tag{2.6}$$

with $\Pi(\mu,\nu) = \{\pi : \pi(\cdot, Y) = \mu, \pi(X,\cdot) = \nu\}$. Moreover, (2.6) admits a duality form, named Kantorovich duality and can be expressed as

$$C(\mu,\nu) = \sup_{(\phi,\psi)\in L^1(\nu)\times L^1(\mu)} \left\{ \int_Y \phi(y)d\nu(y) - \int_X \psi(x)d\mu(x) : \phi(y) - \psi(x) \le c(x,y) \right\}.$$

When we concentrate on the special case where $c(x,y) = d^p(x,y)$ with $d(x,y)$ the Euclidean distance and $p = 1, 2$, we can have more interesting properties and will expand the discussion around it.

2.5.1 Wasserstein-1 and Wasserstein-2 distance

Both the prime and dual formulations provide a static point of view to describe the optimal transport. However, when $X = Y = \mathcal{M} \subset \mathbb{R}^d$ and

$$c(x,y) = \inf_{\gamma} \left\{ \int_0^1 L(\gamma,\dot{\gamma},t)dt : \gamma(0) = x, \gamma(1) = y \right\},$$

we can treat the optimal transport problem with a dynamic point of view rewrite it as an optimal control problem. Given two densities $\rho^0 \sim \mu$, $\rho^1 \sim \nu$ and when $L = \|\dot{\gamma}\|^2$ (that is, we have $c(x,y) = d^2(x,y)$ with $d(\cdot,\cdot)$ the Euclidean distance), we can get the following formula for the problem:

$$W_2(\rho^0,\rho^1) = \inf_{v} \left\{ \left(\int_0^1 \int_{\mathcal{M}} \|v\|_2^2 \rho dx dt \right)^{\frac{1}{2}} : \rho_t + div(v\rho) = 0, \rho(0) = \rho^0, \rho(1) = \rho^1 \right\}, \tag{2.7}$$

where using $\rho(t)$ simply means $\rho(x,t)$ with the x variable omitted (we will keep using this notation afterwards in this thesis), $\rho(x,t)$ is a curve of density function interpolates ρ^0, ρ^1,

and $\|\cdot\|_2$ is the Euclidean 2 norm. Here $W_2(\rho^0, \rho^1)$ is called the Wasserstein-2 distance between ρ^0 and ρ^1. By Hodge decomposition, the optimization can be restricted on the vector field set consisting of the gradient of potential functions

$$W_2(\rho^0, \rho^1) = \inf_v \left\{ \left(\int_0^1 \int_{\mathcal{M}} \|\nabla\Phi\|_2^2 \rho dx dt \right)^{\frac{1}{2}} : \rho_t + div(\rho\nabla\Phi) = 0, \rho(0) = \rho^0, \rho(1) = \rho^1 \right\}.$$

Furthermore in this special case, the optimal solution satisfies the following system of PDEs:

$$\begin{cases} \rho_t(x,t) + div(\nabla\Phi(x,t)\rho(x,t)) = 0 \\ \Phi_t(x,t) + \frac{1}{2}\|\nabla\Phi\|^2 = 0 \end{cases},$$

which gives the geodesics with boundary points as ρ^0, ρ^1 on the probability manifold. The first equation is the continuity equation coming from the constraints of the problem, with the velocity being $\nabla\Phi$. The second equation is a Hamilton-Jacobi-Bellman equation, independent from $\rho(x,t)$, providing the velocity field $\nabla\Phi$.

Meanwhile, when $c(x,y) = d(x,y) = \|x-y\|_p$ where

$$\|x-y\|_p = \left(\sum_{i=1}^d (x_i - y_i)^p \right)^{\frac{1}{p}},$$

we can define the Wasserstein-$(1,p)$ distance as

$$W_1(\rho^0, \rho^1) = \inf_v \left\{ \int_0^1 \int_{\mathcal{M}} \|v\|_p \rho dx dt : \rho_t + div(v\rho) = 0, \rho(0) = \rho^0, \rho(1) = \rho^1 \right\}. \quad (2.8)$$

One thing to notice is that when $c(x,y) = d(x,y)$, the dual form of the problem relies on one single 1-Lipschitz function ψ, and the equation is called the Kantorovich-Rubinstein formula:

$$W_1(\rho^0, \rho^1) = \sup_{\|\psi\|_{Lip} \le 1} \int_{\mathcal{M}} \psi(\rho^1 - \rho^0) dx.$$

As is shown in this equation, W_1 distance is independent from time variable. Actually a

similar property holds for the prime formulation, which is given as below:

$$W_1(\rho^0,\rho^1)=\inf_m\Big\{\int_{\mathcal{M}}\|m\|_p dx\colon div(m(x))+\rho^1(x)-\rho^0(x)=0, m(x)\cdot n(x)=0, \\ \text{for all } x\in\partial\mathcal{M}, n(x) \text{ is normal to } \partial\mathcal{M}\Big\}. \tag{2.9}$$

To show (2.8) is equivalent to (2.9), given an m feasible for (2.9), we define $\rho(t,x)=t\rho^1(x)+(1-t)\rho^0(x)$ and $v(t,x)=m(x)/\rho(t,x)$. Then, $v(t,x)$ is feasible for (2.8) and has the same objective value as m for (2.9). Therefore, we have

$$\inf_v\int_{\mathcal{M}}\int_0^1\|v\|_p\rho(x,t)dtdx\le\inf_m\int_{\mathcal{M}}\|m\|_p dx.$$

On the other hand, by Jensen's inequality, it holds that

$$\int_0^1\|v\|_p\rho(x,t)dtdx\ge\left\|\int_0^1 v\rho(x,t)dt\right\|_p=\|m\|_p,$$

where

$$m(x)=\int_0^1 v(x,t)\rho(x,t)dt.$$

Thus, we have

$$\inf_v\int_{\mathcal{M}}\int_0^1\|v\|_p\rho(x,t)dtdx\ge\inf_m\int_{\mathcal{M}}\|m\|_p dx,$$

from which we conclude that (2.8) and (2.9) identical.

2.5.2 Otto calculus

Wasserstein-2 distance provides a way to define a Riemannian metric on the infinite dimensional manifold $\mathcal{P}(\mathcal{M})$. Given any smooth curve $\rho(x,t)$ satisfying the continuity equation

$$\rho_t+div(\rho v)=0,$$

it is true that under the equivalence relation $\sim$ at the point ρ defined as

$$u \sim v : -div(\rho u) = -div(\rho v),$$

v and ρ_t has a one-to-one correspondence. Also notice that

$$\int_{\mathcal{M}} \rho_t dx = \frac{d}{dt} \int_{\mathcal{M}} \rho dx = 0,$$

we have at any point $\rho \in \mathcal{P}(\mathcal{M})$ the tangent space as

$$T_\rho \mathcal{P}(\mathcal{M}) = \left\{ f : \mathcal{M} \to \mathbb{R} : \int_{\mathcal{M}} f(x) dx = 0 \right\}.$$

Thus, an inner product can be defined in $\mathcal{P}(\mathcal{M})$ as

$$\langle \xi_t(x,0), \eta_t(x,0) \rangle_\rho = \int_{\mathcal{M}} \rho u \cdot v dx, \tag{2.10}$$

where $\xi(x,0) = \eta(x,0) = \rho(x)$ and $\xi_t + div(\xi u) = 0$, $\eta_t + div(\eta v) = 0$. With this, the solution curve of (2.7) can be treated as the geodesics on this probability Riemannian manifold and we call (2.10) the Wasserstein-2 metric. Using this metric, we can calculate the gradient operator. On the one hand, given a functional $\mathcal{F}(\rho)$, we know that $grad\mathcal{F} \in T\mathcal{P}(\mathcal{M})$ with $grad\mathcal{F}_\rho \in T_\rho \mathcal{P}(\mathcal{M})$ and have

$$\begin{aligned}
\langle grad_\rho \mathcal{F}, \mu_t(\cdot,0) \rangle_\rho &= d\mathcal{F}_\rho(\mu_t(\cdot,0)) \\
&= \int_{\mathcal{M}} \mu_t(x,0) \frac{\delta}{\delta \rho} \mathcal{F} dx \\
&= -\int_{\mathcal{M}} div(\mu(x,0)v) \frac{\delta}{\delta \rho} \mathcal{F} dx \\
&= \int_{\mathcal{M}} \mu_t(x,0) v \cdot \nabla \frac{\delta}{\delta \rho} \mathcal{F} dx \\
&= \int_{\mathcal{M}} \rho v \cdot \nabla \frac{\delta}{\delta \rho} \mathcal{F} dx,
\end{aligned}$$

where $\frac{\delta}{\delta\rho}\mathcal{F}$ is calculated by the first variational formula and $\mu_t = -div(\mu v)$ with $\mu(\cdot, 0) = \rho$.

On the other hand, letting $\eta_t = -grad\mathcal{F}$ where $\eta_t + div(\eta u) = 0$, we have $grad\mathcal{F} = div(\eta u) = -div(\eta(-u))$. Therefore, with (2.10),

$$\langle grad\mathcal{F}, \mu_t(\cdot, 0)\rangle_\rho = -\int_{\mathcal{F}} \rho u \cdot v dx$$

Since the above calculation holds at all $\rho \in \mathcal{P}(\mathcal{M})$, we have the representation of $grad\mathcal{F}$ to be $-\nabla\frac{\delta}{\delta\rho}\mathcal{F}$ and can rewrite the gradient flow induced by (2.10) as

$$\rho_t = div\left(\rho\nabla\frac{\delta}{\delta\rho}\mathcal{F}\right), \tag{2.11}$$

with $\mathcal{F}$ being the Lyapunov function since

$$\frac{d}{dt}\mathcal{F}(\rho) = d\mathcal{F}(\rho_t) = \langle grad\mathcal{F}, \rho_t\rangle = -\langle grad\mathcal{F}, grad\mathcal{F}\rangle \leq 0$$

As a special case, if the functional is given as the free energy

$$\mathcal{F}(\rho) = \int_{\mathbb{R}^d} V(x)\rho(x)dx + \frac{1}{2}\iint_{\mathbb{R}^d\times\mathbb{R}^d} W(x,y)\rho(x)\rho(y)dxdy + \beta\int_{\mathbb{R}^d}\rho(x)log\rho(x)dx,$$

by first variational formula, we have the differential of $\mathcal{F}$ to be

$$\frac{\delta}{\delta\rho}\mathcal{F}(\rho(x)) = V(x) + \int_{\mathbb{R}^d} W(x,y)\rho(y)dy + \beta\log\rho(x) + C,$$

where C is a constant and the gradient flow is

$$\rho_t(x,t) = div\left(\rho(x,t)\nabla\left(V(x) + \int_{\mathbb{R}^d} W(x,y)\rho(y,t)dy\right)\right) + \beta\Delta\rho(x,t),$$

which is the Fokker-Planck equation. Thus, Fokker-Planck equation is a gradient flow

under the Wasserstein-2 metric. One can show that it tends to the Gibbs' distribution

$$\rho^*(x) = \frac{1}{K} \exp\left(-\frac{V(x) + \int_{\mathbb{R}^d} W(x,y)\rho^*(y)dy}{\beta} \right),$$

where

$$K = \int_{\mathbb{R}^d} \exp\left(-\frac{V(x) + \int_{\mathbb{R}^d} W(x,y)\rho^*(y)dy}{\beta} \right) dx.$$

2.5.3 Wasserstein-2 metric on finite graphs

To do computation of (2.11) on a discrete finite graph $G = (V, E)$, where V is the node set and E is the edge set illustrating the connection relation among edges, we need a discrete version of the equation. Here, G can have arbitrary structure, including a discrete grid mesh discretized from the continuous space. To this end, we start from the definition of discrete Wasserstein-2 metric. Same as the continuous space, denoting $n = |V|$, the probability manifold on G is

$$\mathcal{P}(G) = \left\{ \rho = (\rho_i)_{i=1}^{|V|} \in \mathbb{R}^n : \sum_{i=1}^{n} \rho_i = 1 \right\}$$

and the tangent space at ρ is

$$T_\rho \mathcal{P}(G) = \left\{ \sigma = (\sigma)_{i=1}^{n} \in \mathbb{R}^n : \sum_{i=1}^{n} \sigma_i = 0 \right\}.$$

For a potential function $\Phi = (\Phi_i)_{i=1}^n$, we introduce a discrete divergence operator with respect to $\rho \in \mathcal{P}(G)$:

$$div(\rho \nabla \Phi) = \left(-\sum_{j \in N(i)} (\Phi_i - \Phi_j) g_{ij}(\rho) \right)_{i=1,2,\cdots,n},$$

where $N(i) = \{j : (i,j) \in E\}$ is the neighborhood of i. Also we define an inner product of $\nabla\Phi$ at ρ

$$\langle \nabla\Phi, \nabla\Phi \rangle_\rho = \frac{1}{2} \sum_{(i,j)\in E} (\Phi_i - \Phi_j)^2 g_{ij}(\rho).$$

In the above two equations, g_{ij} satisfies

$$g_{ij}(\rho) = g_{ji}(\rho) \text{ and } \omega_{ij} \min(\rho_i, \rho_j) \leq g_{ij}(\rho) \leq \omega_{ij} \max(\rho_i, \rho_j),$$

where $\omega_{ij} > 0$ are the weights assigned on E. With this operator, we define an equivalence relation in $\mathbb{R}^n$ with quotient space as

$$\mathbb{R}^n / \sim = \{[\Phi] : (\Phi_i)_{i=1}^n \in \mathbb{R}^n\}, \text{ where } [\Phi] = \{(\Phi_1 + c, \cdots, \Phi_n + c) : c \in \mathbb{R}\}.$$

Then for each σ, by Hodge decomposition, we can have a one-to-one correspondence τ defined as $\sigma = \tau([\Phi])$ where $\tau([\Phi]) = -div(\rho\nabla\Phi)$. And the metric tensor can be defined as below:

$$g_\rho(\sigma_1, \sigma_2) = \frac{1}{2} \sum_{(i,j)\in E} g_{ij}(\rho)(\Phi_i^1 - \Phi_j^1)(\Phi_i^2 - \Phi_j^2).$$

The discrete Wasserstein-2 distance is given in the following format:

$$W_2(\rho^0, \rho^1) = \inf_{\Phi} \left\{ \left(\int_0^1 \langle \nabla\Phi, \nabla\Phi \rangle_\rho dt \right)^{\frac{1}{2}} : \rho_t + div(\rho\nabla\Phi) = 0, \rho(0) = \rho^0, \rho(1) = \rho^1 \right\}$$

Now given a functional $\mathcal{F}(\rho)$ defined on $\mathcal{P}(G)$, we have that

$$\mathcal{F} : \mathbb{R}^d \to \mathbb{R}.$$

To discretize the gradient flow (2.11), we specifically choose g_{ij} in the upwind direction of

the flow and set the metric tensor as:

$$g_{ij} = \begin{cases} \omega_{ij}\rho_i & F_i(\rho) > F_j(\rho), j \in N(i) \\ \omega_{ij}\rho_j & F_i(\rho) < F_j(\rho), j \in N(i) \\ \frac{\omega_{ij}}{2}(\rho_i + \rho_j) & F_i(\rho) = F_j(\rho), j \in N(i) \end{cases},$$

where $F_i(\rho) = \frac{\partial}{\partial \rho_i}\mathcal{F}(\rho)$. Then the discrete gradient flow of $\mathcal{F}$ induced by discrete Wasserstein-2 distance is

$$\rho_t = -L(\rho)\nabla_{L^2}\mathcal{F}(\rho) \tag{2.12}$$

where $\nabla_{L^2}\mathcal{F}$ is the standard L^2 gradient for $\mathcal{F}$ and the $n \times n$ matrix $L(\rho)$ is a projection (the weighted Laplacian matrix) defined as

$$L(\rho)_{ij} = \begin{cases} g_{ij} & (i,j) \in E \\ 0 & (i,j) \notin E \end{cases}.$$

Thus, (2.12) can be also written in the following entry-wise format:

$$\dot{\rho}_i = \sum_{j \in N(i)} \omega_{ij}\rho_j(F_j(\rho) - F_i(\rho))_+ - \sum_{j \in N(i)} \omega_{ij}\rho_i(F_i(\rho) - F_j(\rho))_+,$$

where $(\cdot)_+ = \max(\cdot, 0)$.

CHAPTER 3

PATH PLANNING IN UNKNOWN ENVIRONMENTS

3.1 Introduction

In this chapter, we study the path planning for a group of robots in an unknown environment, that is, a set of robots with given initial configurations finds a feasible path to the target configurations while avoiding any collisions with obstacles or violation of constraints. In our considerations, we assume that the number of robots is fixed, the obstacles can only be detected and memorized when they are within a detectable distance along the motion of robots. Compared to the path planning in known environments, there are several significant challenges when the problem is posed in unknown environments. First of all, re-planning while moving deems to be inevitable when a planned path is blocked by newly detected obstacles. This raises the concern whether the robots may move in circular loops, and never reach the target even there exist feasible paths. How to avoid moving in repetitive patterns becomes necessary for any reasonable algorithms. Secondly, the configuration space may be in a high dimensional subspace of $\mathbb{R}^n$, such as those in multi-agent systems. Using grid-like discretization often leads to intractable computations. In this case, working with graphs is a viable option to reduce the computation burden. However, the cost can still be high if the graph has to span everywhere in the high dimensional space. Thirdly, in spite of the size of robots, there may be narrow pathways between obstacles, which poses significant hurdles to identify them in the search process. How to ensure an algorithm succeed in this kind of environment is an important consideration in practice. In addition, for problems with unknown environments, optimality is only meaningful in the currently known environments. Hence, one may have to accept locally optimal, or even simply feasible, solutions in some cases.

There exists an extensive literature for path planning, many of them have achieved remarkable success. For example, the well-known Probabilistic Road Map (PRM) generates a random graph that does not intersect with obstacles and then finds a path on the graph to connect the initial and target configurations [4, 5, 6, 7]. PRM guarantees to create a connection between the initial and target configurations when the graph is dense enough in the configuration space. Many improvements of PRM have been reported in the past decades, see [8, 9, 10, 11] for details.

The Rapidly-exploring Random Tree (RRT) is another influential algorithm that has been found efficient in many applications [12]. It creates a tree structure rooted at the initial configuration, making sure that all the vertices are connected to the initial one. At each step, the algorithm picks a configuration in the space randomly, and checks whether it can be added to the tree following certain criteria. This is continued until the target or a configuration close enough to the target is included in the tree. A key to the success of the algorithm is the randomness in choosing configurations. Biased choices may cause trapped path at local minima. This algorithm has been adopted to path finding in unknown environments [13, 14] and has recent improvements such as RRT* [15], and the ones reported in [16, 17].

The Artificial Potential Field (APF) assigns the robot a positive charge while the target configuration a negative one [18] . This drives the robot towards the target. To avoid collisions with the obstacles, APF sets the obstacles with positive charges that repel the robot. Since it is a local gradient method, APF is friendly to high dimensional problems. However, a major limitation of the original APF is the creation of unnecessary local minima due to the presence of obstacles, which may fail the algorithm. In recent years, there are reported improvements of APF, see [19, 20, 21, 22] and the references therein.

The family of Bug Algorithms, starting from the original Bug0, Bug1 and Bug2 [23, 24] to the later developments, such as TangentBug [25], DistBug [26] and many other variants, adopt two basic modes as their design principle: motion-to-goal mode and boundary-

following mode. They are powerful tools, with theoretical convergence guarantees, especially suitable for path planning in unknown environments in 2 dimensional working space. Some recent survey and performance comparison studies can be found in [27, 28].

In addition, widely known graph based methods, like A^* [29], D^* [30], Focused D^* [31] and D^* lite [32] can be used for path planning in both known and unknown environments [33]. When applied to the unknown environment problems, they often require to cover the entire region by discrete lattice grids, on which the algorithms are performed to find optimal paths. In literature, there are other types of algorithms such as genetic algorithm [34, 35], evolutionary programming [36], fuzzy logic [37, 38], neural network [39], network simplex method [40], method of evolving junctions [2], fast marching tree [41], and a few hybrid approaches that combine different methods [42, 43], and also many more swarm strategies for multi-agent systems in recent years [44, 45, 46].

In this chapter, we present a potential guided, graph-based pathfinding method inspired by the evolution of Fokker-Planck Equation (FPE) in optimal transport theory. Our algorithm has the following features:

1. The algorithm is a graph based deterministic procedure with guaranteed convergence property, meaning that the algorithm stops in a finite number of steps, and returns a feasible path if there exists one. Thus, the algorithm is complete. We would highlight that the convergence of our algorithm is deterministic, in contrast to the asymptotic convergence results shared by many methods using randomness.

2. The algorithm always stops in finite steps. If it does not find a feasible path, one concludes that from initial to target configurations, there does not exist a feasible trajectory such that there is a tubular region, centered at the trajectory with a small radius, not intersecting with obstacles. The lower bound of the radius for the tubular region is proportional to the step size used in graph generation.

3. The path found by the algorithm is locally optimal in the known environment up to

the current location of the robots.

4. The graph generated by the algorithm has a tree structure growing linearly with respect to the dimension of the configuration space. Together with the dimension reduction techniques proposed to rapidly escape the local traps, the algorithm can efficiently handle high dimensional problems.

5. The algorithm only explores a limited region defined by the solution of FPE, even when the obstacles are not known *a priori*.

It should be noted that optimal transport theory has been considered in several recent studies for path planning. In [47], they treat the multi-agents system to be a distribution and calculate the optimal transport mapping from initial to target, using the primal formulation. [48] transform the optimal transport problem to the Kantorovich duality and design a gradient flow procedure to find the optimal. Our method directly uses the potential information and the graph construction is guided by the evolution of FPE.

In the next section, we present the details of the algorithm with the finite step stopping property. In Section 3.3, we show some numerical examples to illustrate the performance in both low and high dimensional configuration spaces. Section 3.4 gives strategies for dimension reduction near local minima to further lower the computational cost. In Section 3.5 the relationship between the algorithm and optimal transport theory is discussed. The convergence proof is given in Section 3.6. We end this chapter with a brief conclusion in the last section.

3.2 Algorithm

Let the configuration space Ω be a bounded connected domain in $\mathbb{R}^n$. We assume that the robots can alter configurations freely in Ω as long as the change does not violate the required constraints. There are two types of constraints we consider in this chapter. One is the constraints known in advance, for example, two robots can't be too close or too far

away from each other in the multi-agent system. We denote those constraints by

$$\phi = (\phi_1, \phi_2 \cdots, \phi_{k_1}) : \Omega \longrightarrow \mathbb{R}^{k_1},$$

and a configuration $x \in \Omega$ does not satisfy the constraints when $\phi_i(x) < 0$ for some $i \in \{1, \cdots, k_1\}$. The other type of constraints is given by unknown environments, such as unknown obstacles. We represent them by

$$\psi = (\psi_1, \psi_2 \cdots, \psi_{k_1}) : \Omega \longrightarrow \mathbb{R}^{k_2},$$

and $\psi_i(x) < 0$ for some $i \in \{1, \cdots, k_2\}$ means the constraints are violated. We emphasise that $\psi(x)$ can only be detected if robots are close enough to the obstacles. This implies that the knowledge of $\psi(x)$ must be updated dynamically while the robots are in motion. For the simplicity in discussion, we assume that both ϕ and ψ are continuous.

To illustrate the setups, we give a single robot example in Figure 3.1. The configuration space is a square, all the gray bars are the obstacles that the robot cannot collide with. The light gray bars in the figure are obstacles undetected. Like in the second picture in Figure 3.1, if the robot moves horizontally but not too far away from its initial configuration, there is no detected obstacle. As the robot moves, more and more obstacles are recognized. Our goal is finding a path from the initial configuration x_s (red diamond in Figure 3.1) to the target configuration x_t (red circle in Figure 3.1). More precisely, we want to find a feasible path

$$\gamma(t) : [0, T] \to \Omega,$$

satisfying $\phi(\gamma(t)) \geq 0$ and $\psi(\gamma(t)) \geq 0$ for all $t \in [0, T]$, such as the red path in Figure 3.1, while $\psi(x)$ is updated with newly detected obstacles as $\gamma(t)$ changes.

To describe the dynamical change of the unknown constraints while moving along a path, we mark a configuration x as part of the detected obstacles if ($\psi_i(x) < 0$) and x is

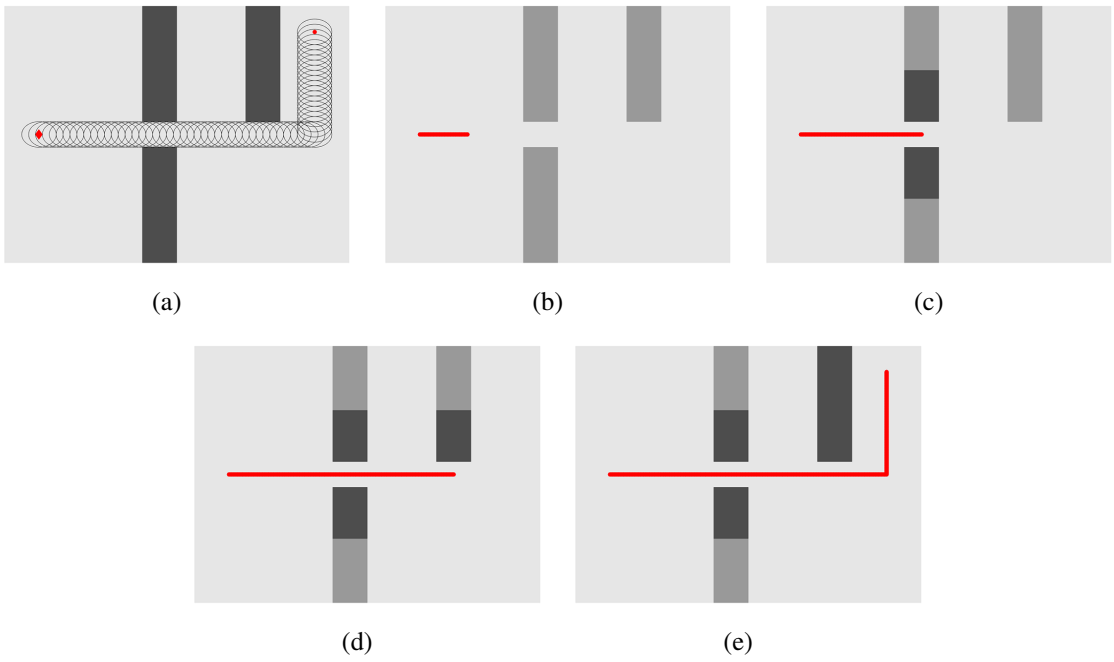


Figure 3.1: Environment, Moving and Obstacle Free Tube: The environment obstacles are the light and dark gray tubes (light as undetected and dark as detected). In (a), the red diamond and circle are the start and target configurations of the robot and the shaded tubular region is the obstacle-free region $\mathcal{T}$. (b)-(e) are the robot moving process along a certain path in chronological order.

within distance R to the current configuration of the robot. To be precise, we define

$$\hat{\psi}(x,\gamma,t) = \begin{cases} \psi(x) & \text{if } \|x-\gamma(\tau)\| \leq R \text{ for some } \tau \leq t \\ 0 & \text{otherwise} \end{cases} \tag{3.1}$$

as the detected part of the environment along the path.

For the convenience of discussion, we assume there exists at least a feasible path connecting the initial and target configurations, and this feasible path is contained in a tubular obstacle-free region $\mathcal{T}$ with radius L as is shown by the shadow part in the first picture in Figure 3.1. This assumption is a technique requirement that is used for the proof of the convergence and can be rewritten as the following equation

$$\sup_{\gamma\in\Gamma} \inf_{t\in[0,T]} \sup_{r\geq 0} \{r : B(\gamma(t),r) \cap \mathcal{O} = \emptyset\} = L > 0, \tag{3.2}$$

where

$$\mathcal{O} = \{x \in \Omega : \psi_i(x) < 0 \text{ for some } i \in \{1, \cdots, k_1\} \text{ or } \phi_i(x) < 0 \text{ for some } i \in \{1, \cdots, k_2\}\},$$

which is an open set, $B(x, r) = \{y \in \Omega : \|x - y\| < r\}$ is also open. We denote $S(x, r) = \{y \in \Omega : \|x - y\| = r\}$ as the boundary of $B(x, r)$ and $\partial\mathcal{O}$ as the boundaries of $\mathcal{O}$, separating the constrained regions from the feasible regions.

Let us define the set of all possible paths from x_s to x_t in the full time interval $[0, T]$ as

$$\Gamma = \{\gamma : \gamma(0) = x_s, \gamma(t) = x_t, \forall t \geq T_0, \text{ for some } T_0 \leq T, \gamma \cap \mathcal{O} = \emptyset\} \tag{3.3}$$

Then our dynamical path planning algorithm in the unknown environment is given as the following:

Algorithm 2: Path Planning in Unknown Environment

Data: initial configuration x_s, target configuration x_t, initially known constraints $\mathcal{O}$

Current configuration $x_c = x_s$

Current known constraints $\mathcal{O}_c = \mathcal{O}$

while $x_c \neq x_t$ **do**

  Graph Generating: Generate a connected graph G containing x_c, x_t with all edges and vertices not in $\mathcal{O}_c$

  Path Finding: Find a (shortest) path γ on G from x_c to x_t

  Environment Updating: Moving along γ while updating $\mathcal{O}_c$, if γ is blocked by $\mathcal{O}_c$, stop at x near the block point, otherwise let $x_c = x_t$

end

In the remaining part of this section, we discuss, assisted with examples, the three major steps in details.

3.2.1 Graph generating

The first step is to generate a graph $G = (V, E)$, where V is the vertex set and E is the edge set, connecting the current configuration x_c ($x_c = x_s$ for the first graph generation) and the target x_t with currently known environment. The vertices are configurations in Ω while the edge (u, v) linking $u, v \in V$ is the straight line segment between u and v. Meanwhile, we would like to create the graph satisfying two properties: 1) the graph does not violate any known constraints; 2) the graph cannot contain too many vertices due to the computation complexity concern in the high dimensional case. To achieve these goals, we introduce a convex potential function $p(x)$, admitting a unique global minimizer x_t, to help us choose the vertices. We select an n-dimensional orthonormal basis N (here n is the dimension of Ω) to determine the directions which are used to add new vertices to V. For simplicity, we take $p(x) = \|x - x_t\|$, the distance to the target, as the potential, and the standard coordinate axes $N = \{e_j\}_{j=1}^{n}$ are used as the orthonormal basis in this chapter.

At the first generating step, we let $V = \{x_c\}$ and $E = \emptyset$. In each step afterward, a vertex $v \in V$ with the lowest potential is chosen. We pick $2n$ new points $\{v_i\}_{i=1}^{2n}$ along the orthonormal basis N originated at v, with distance l to v, and use them as the candidates to expand V (first figure in Figure 3.2). Before adding those points into the vertex set, we first delete all candidates that violate the currently known constraints ($\phi_k(v_i) < 0$ or $\hat{\psi}_j(v_i, \gamma, T_0) < 0$ for some $k \in \{1, \cdots, k_1\}$ and $j \in \{1, \cdots, k_2\}$, γ is the previous trajectory of the robots). For example, the robot shown in the left plot in Figure 3.2 stops at the red diamond position and generates four points around it. Among them, the point in the obstacle is removed (right picture in Figure 3.2). Next, we delete vertices whose edges violate the constraints as shown in Figure 3.3. In this case, there exists a point $x \in (v_i, v)$ such that x is not in the feasible region. In addition, to avoid repeating vertices, we remove those already included in V from the candidate list, as shown in Figure 3.4. After these deleting steps, we add all remaining candidates, and their associated edges, to V and E respectively. This process is repeated until the target x_t is within a small neighborhood of

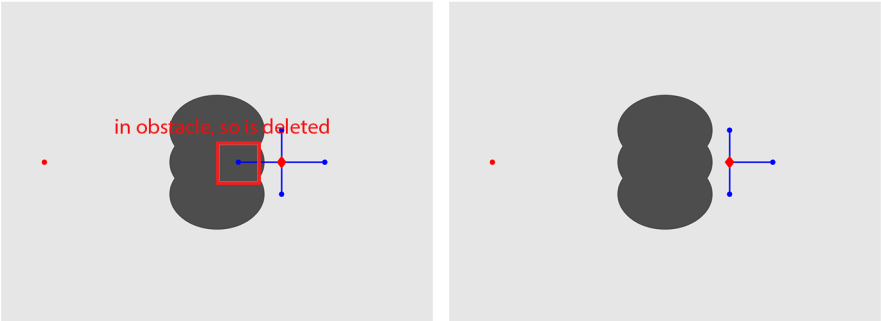


Figure 3.2: graph generating steps: delete nodes in obstacles

a vertex in V. For example, the final graph after several iterations is plotted in the right figure of Figure 3.4.

To summarize, given the previous trajectory of robots γ ($\gamma = x_s$ as default), we let the current constraints be

$$\mathcal{O}_c = \Big\{ x \in \Omega : \hat{\psi}_i(x, \gamma, T_0) < 0 \text{ for some } i \in \{1, \cdots, k_2\} \\ \text{or } \phi_i(x) < 0 \text{ for some } i \in \{1, \cdots, k_1\} \Big\}.$$

The graph generating procedure can be written in the following algorithm (Algorithm 3), and we define x is the ancestor of y if when x is picked to generate nodes as the lowest potential node, y is added to the vertex set as newly generated node.

Remark 1. *The choice of the generating radius l can be arbitrary, although L and l must satisfy an inequality to have the convergence guarantee theoretically (as is shown in Section 3.2.4). Larger l leads to fewer vertices in V while smaller l gives a finer search in Ω. For simplicity, we treat those obstacles with distance less than l to be a single obstacle by ignoring the gaps among them in our theoretical analysis. In practice, the graph generation can still create nodes passing through the gap between obstacles with distance less than L or even l in our experiments.*

Algorithm 3: Graph Generation

Data: The initial configuration x_s, target configuration x_t, the potential function p, currently known environment $\mathcal{O}_c$, graph generating radius l and a set of orthonormal basis N

$V = \{x_s\}, Q = V, E = \emptyset$
while $t \notin V$ **do**
  $point\_add = False$
  while *not* $point\_add$ **do**
    $v = \arg\min_{x \in V} p(x)$
    if $p(v) < +\infty$ **then**
      $K = \{q : q = v \pm l \times y, y \in N, (v, q) \cap \mathcal{O}_c = \emptyset, q \notin \mathcal{O}_c\}$
      $K = K \backslash V$
      $V = V \cup K$
      $E = E \cup \{(v, q) : q \in K\}$
      if $K \neq \emptyset$ **then**
        $point\_add = True$
        $p(v) = +\infty$
      end
      for $q \in K$ **do**
        if $\|q - x_t\| \leq L$ *and* $(q, x_t) \cap \mathcal{O}_c = \emptyset$ **then**
          $V = V \cup \{x_t\}$
          $E = E \cup \{(q, x_t)\}$
        end
      end
    else
      return $G = \emptyset$
    end
  end
end
return $G = (V, E, p)$

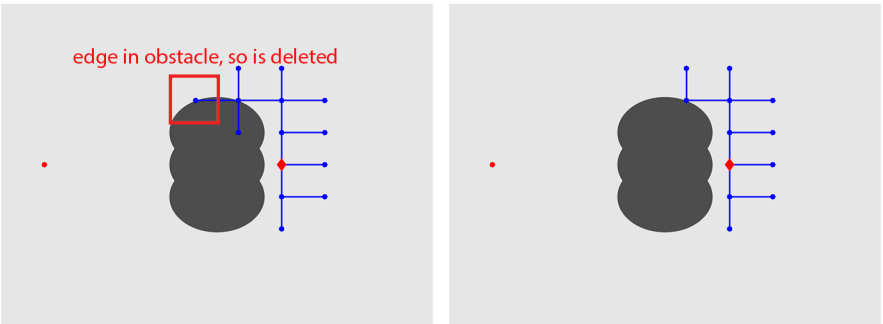


Figure 3.3: graph generating steps: delete nodes cannot be linked to the base node

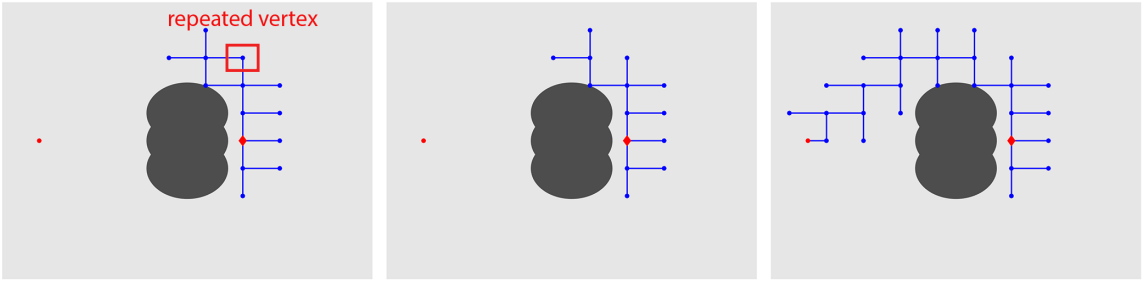


Figure 3.4: graph generating steps: delete repeated nodes (left two) and final graph (right one)

3.2.2 Path finding

After generating the graph $G = (V, E, p)$, the next step is to find a feasible path moving from the current configuration to the target using only vertices and edges on the graph. Our goal is to minimize the total travel distance. The graph generated by Algorithm 3 has the following property:

Proposition 1. *There exists a unique path from initial to target configurations over the generated graph G. And if the path is denoted by $\{x_i\}_{i=0}^{q} \subset V$ with*

$$x_0 \to x_1 \to \cdots \to x_q = x_f,$$

x_i is the ancestor of x_{i+1}.

Proof. The existence of a path from initial to target is natural by the construction of the graph. Also, notice that if two nodes share an edge, one of them is the ancestor of the other one, which is determined by the graph construction strategy. Thus, we know that x_0 is the ancestor of x_1. And by induction, we assume that x_i is the ancestor of x_{i+1}, then if x_{i+1} is not the ancestor of x_{i+2}, it implies that x_{i+2} is the ancestor of x_{i+1}, which means that x_{i+i} has two ancestors. This is a contradiction with the graph generation strategy (in each graph generating step, we delete all candidate nodes that are already generated in previous steps). Therefor, in the path, x_i is the ancestor of x_{i+1}.

For the uniqueness, first we notice that the algorithm stops once there is an edge linking

the target x_f, from which we conclude that x_f has unique ancestor. And by Algorithm 3, it is true that every node except the initial x_0 has unique ancestor as well. Now assume we have two paths γ, ξ, and denote the nodes they pass as $\{y_i\}_{i=0}^{m_1}, \{z_i\}_{i=0}^{m_2}$ where $y_0 = z_0 = x_0$ and $y_{m_1} = z_{m_2} = x_f$. Following this, we have $y_{m_1-1} = z_{m_2-1}$ since y_{m_1-1}, z_{m_2-1} are ancestors of x_f. Again by induction, it is true that $m_1 = m_2$ and $y_i = z_i$. Thus, $\gamma = \xi$ and the uniqueness is proven. □

Since if there is an edge between two nodes, by the graph generation strategy, one of the node must be the ancestor of the other one. This suggests a simple strategy to identify the path: From the target configuration, we simply back trace the ancestor of each node in the path until reaching starting configuration x_c.

Other algorithms can be applied to find the path as well. For example, we can define the distance of the edge e_{ij} linking vertices v_i, v_j as

$$k_{ij} = len((i,j)) = \|v_i - v_j\|.$$

Then the well-known Dijkstra method, or its improvements, can be used to obtain the path with computational complexity $O(|E| + |V| \log |V|)$ where $|V|$ is the number of vertices and $|E|$ is the number of edges [49].

Another way is to assign each edge distance 1 which is equivalently to introduce the modified adjacency matrix $K = (k_{ij})$ on the graph G, where

$$k_{ij} = \begin{cases} 1 & \text{if } (i,j) \in E \\ \infty & \text{otherwise.} \end{cases}$$

Then the Breadth First Search (BFS) can be used to find the path with the complexity $O(|E| + |V|)$ [50, 51], which is faster than Dijkstra. Other graph-based path planning algorithms, such as A^*, D^* or D^* lite, can be used too.

It is worth mentioning that if we assume the path has Λ nodes, the suggested back-

tracing approach is of complexity $O(\Lambda)$. While the generated graph has at least $O(n\Lambda)$ nodes. Obviously the complexities of BFS and Dijkstra methods are higher than our back-tracing strategy.

3.2.3 Environment updating

While robots move along a path γ_i in the configuration space, the knowledge of constraints is updated at the same time by (3.1). We let

$$\gamma_i : [0, T_0] \to \Omega$$

be the current path, and if a point on the path intersects the boundary of the constrained region, the motion stops at a point before arriving the intersection.

To be more precise, let us denote the environment update at each time step as

$$\mathcal{O}_c^t = \mathcal{O}_c \cup \left\{ x \in \Omega : \hat{\psi}_j(x, \gamma_i, t) < 0 \text{ for some } j \in \{1, \cdots, k_2\} \right\}.$$

If the path is found activating constraints while moving at time T_s, i.e. $\gamma_i \cap \mathcal{O}_c^{T_s} \neq \emptyset$, we define

$$T_b = \inf\{t : \gamma_i(t) \in \mathcal{O}_c^{T_s}\}$$

as the first intersection time. Then $\gamma(T_b)$ must be on the boundary of $\mathcal{O}_c^{T_s}$, i.e. $\gamma(T_b) \in \partial\mathcal{O}_c^{T_s}$. When this happens, we can always pick a stopping time $T_i \leq T_b$ such that the distance from $\gamma(T_i)$ to the nearest obstacle is smaller than the detection radius R. Afterwards, we update $\mathcal{O}_c = \mathcal{O}_c^{T_i}$, assign the initial configuration as $x_c = \gamma(T_i)$ and go back to the graph generating step. Each time a new path γ_i is produced when the current path is blocked. We collect all paths produced in Algorithm 2 as $\{\gamma_i\}_{i=1}^m$, and their stopping time set as $\{T_i\}_{i=1}^m$. From the choices of stopping time, we may require that there exists a positive constant q satisfying $q < R$, and for all $\epsilon > q$, $B(\gamma_i(T_i), \epsilon)$ has non-empty intersection with $\mathcal{O}_c$ for

every $i = 1, \cdots, m$. Such selected stopping time set satisfies the following property

$$\sup_i \inf_\epsilon \left\{ \epsilon : B(\gamma_i(T_i), \epsilon) \cap \mathcal{O}_c^{T_i} \neq \emptyset \right\} = q < R, \tag{3.4}$$

in which the detectable region at configuration x, using (3.1), is defined as a closed set by

$$\bar{B}(x, R) = \{ y \in \Omega : d(x, y) \leq R \}$$

It is worth noticing that q is independent from m. Each time, we can simply let robots stop at the position that has a distance of $R/2$ to the obstacles when the path is blocked. In this setup, $q = R/2 < R$ and (3.4) naturally holds. In general, the stop position can be different. As long as (3.4) is satisfied, the finite-step convergence property is guaranteed.

3.2.4 Convergence and complexity

The proposed algorithms terminate in finite steps with guaranteed convergence, which is stated in the following main theorems.

Theorem 3.2.1. *Assuming that (3.2) is true and*

$$l < \frac{2L}{\sqrt{n}},$$

where n is the dimension of Ω, the graph generation algorithm (Algorithm 3) stops in finite steps. That is, the loop in the algorithm terminates in finite iterations, the generated graph $G = (V, E, K, p)$ is connected and has a finite number of vertices $|V| < \infty$. Furthermore, $x_s, x_t \in V$ if $\Gamma \neq \emptyset$.

Theorem 3.2.2. *Let $\{\gamma_i\}_{i=1}^m$ be the paths produced by Algorithm 2 with $\{T_i\}_{i=1}^m$ being the stopping time set. If the assumptions in Theorem 3.2.1 and (3.4) hold, then $m < \infty$.*

Theorem 3.2.1 shows that, given the currently known environment, the graph generating procedure stops in finite steps. Theorem 3.2.2 tells that our algorithm breaks the loop

in Algorithm 2 in finite steps. The two theorems together ensure that Algorithm 2 is convergent in finite steps and guarantees a feasible path with the condition (3.2). Therefore, the algorithm is complete.

Furthermore, if the configuration space Ω is of dimension n, there are at most $2n$ new points generated at each step in Algorithm 3, hence the growth rate for the size of the graph V is $O(n)$ at each iteration. The complexity of the Updating Environment step relies on the techniques used to detect the environment, so we do not consider it here. Overall, considering the complexity of the pathfinding algorithms (Dijkstra or BFS) discussed in Section 3.2.2, the proposed algorithms are scalable to high dimensional problems, because it stops in finite steps and the growth of the graph is controlled linearly. This feature is illustrated by our numerical examples presented in the next section.

3.3 Numerical Examples

We set the working space to be $[0, 1] \times [0, 1]$ in all examples and denote the graph generating radius as l. In this section, we show various low-dimensional experiments to give a basic impression on how the algorithm works, followed by several high dimensional cases with different environments. In all examples, the start configurations are always marked as red diamonds while the targets are the red circles. And the potential function is taken as the Euclidean distance from any point x to the target x_t, i.e. $p(x) = \|x - x_t\|$ where $x, x_t \in \Omega \subset \mathbb{R}^n$.

3.3.1 Low dimensional cases

The first example is one robot moving in an unknown environment (Figure 3.5). The configurations are the physical locations of the robot so this is a two-dimensional problem. We take $l = 0.03$ in our graph generating algorithm. Initially, the robot is at the top right corner. It only has the knowledge of a few nearby obstacles at the beginning, and other obstacles are not known. Hence, the graph expands towards the target greedily until reaching

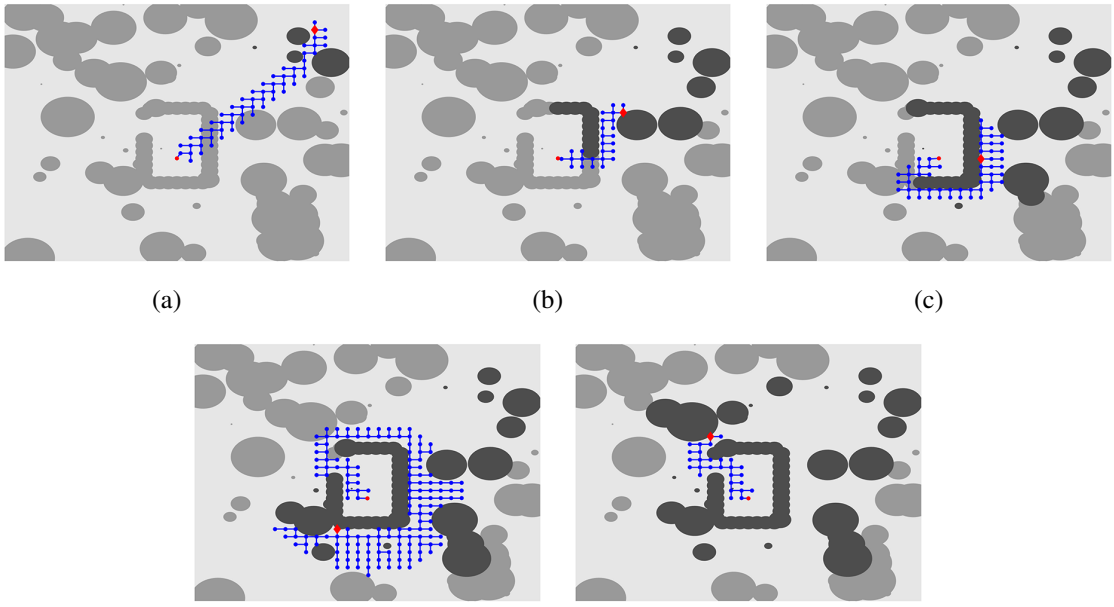
the destination as shown in Figure 3.5(a). The first path is found by BFS on this graph, shown in Figure 3.6(a). However, while moving, the robot detects that the path is blocked. It stops before reaching the obstacle boundary and starts a new round of graph generating, pathfinding and environment updating steps. During the process, the robot generates several graphs (Figure 3.5(b,c,d)) and updates the environment while moving along the corresponding paths as shown in Figure 3.6(b,c,d), all of which fail to reach the destination. In the end, it generates a graph (Figure 3.5(e)) and finds a path (Figure 3.6(e)) to the target. The complete path from initial to target is provided in Figure 3.6(f).

The set-ups of the next example are all the same except the initial configuration. The generated graphs are depicted in Figure 3.7(a-e) in time order and the corresponding paths are in Figure 3.8(a-e) while the complete path is shown in Figure 3.8(f). Despite of the difference in the initial configurations, the algorithm gives similar paths (Figure 3.6(f) and 3.8(f)). In the next experiment, we keep the settings used in the second example, but enlarge the generating radius l from 0.03 to 0.05. By doing so, the robot can no longer move into the central box from the top left corner as it does in the first two examples (Figure 3.6(e) and 3.8(e)). Instead, it moves down and finds a different way to the destination. This path reaches higher potential area than the previous paths. The graphs for this example are depicted in Figure 3.9 and paths are in Figure 3.10 respectively.

To conclude the lower dimensional cases, we display the graphs and paths produced by Algorithm 2 for a different environment in Figure 3.11 and 3.12. Similar behaviors can be observed in those pictures.

3.3.2 High dimensional cases

In the next few examples, we calculate the paths for several multiple-agent systems. In addition to the constraints imposed by the obstacles, we also enforce that the robots cannot be too close or too far away from each other. In our examples, we set that any two robots must keep their distance between 0.03 and 0.13 when moving in the unknown environment.



(a) (b) (c)

(d) (e)

Figure 3.5: The graph produced in the one robot case with generating radius $l = 0.03$ and light (dark) gray the undetected (detected) obstacles. The graph expands greedily towards the target. If obstacles are on the greedy direction, it searches around the obstacles and generates new nodes with potential as low as possible. (a)-(d) are graphs that cross undetected obstacles so the robot stops while moving on those graphs. With enough environment knowledge, (e) is a graph containing a true feasible path from the current initial configuration and the target.

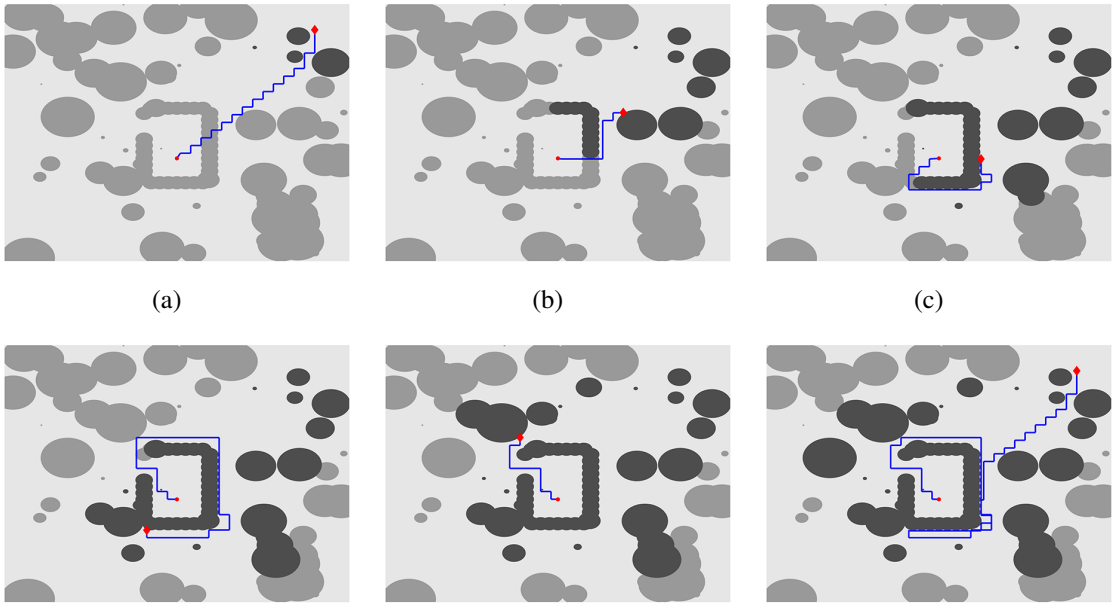


Figure 3.6: The paths calculated based on the results in Figure 3.5. (a)-(d) are middle steps that the robot stops because of the newly detected obstacles while moving and (e) is the path on which the robot get to the target. (f) gives the complete path of the robot.

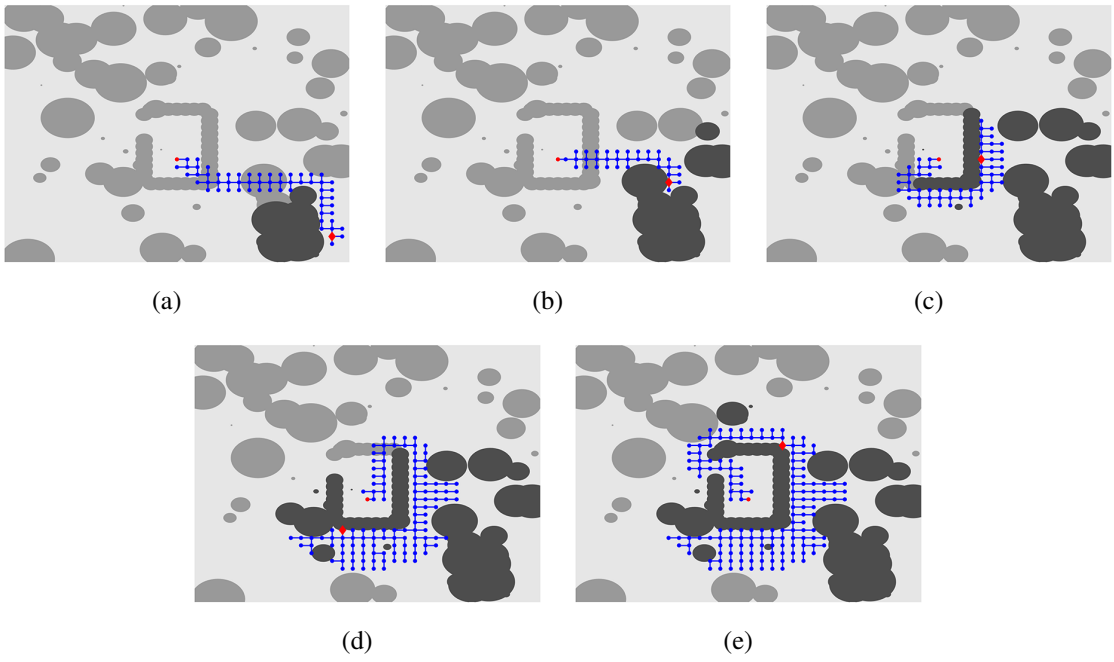
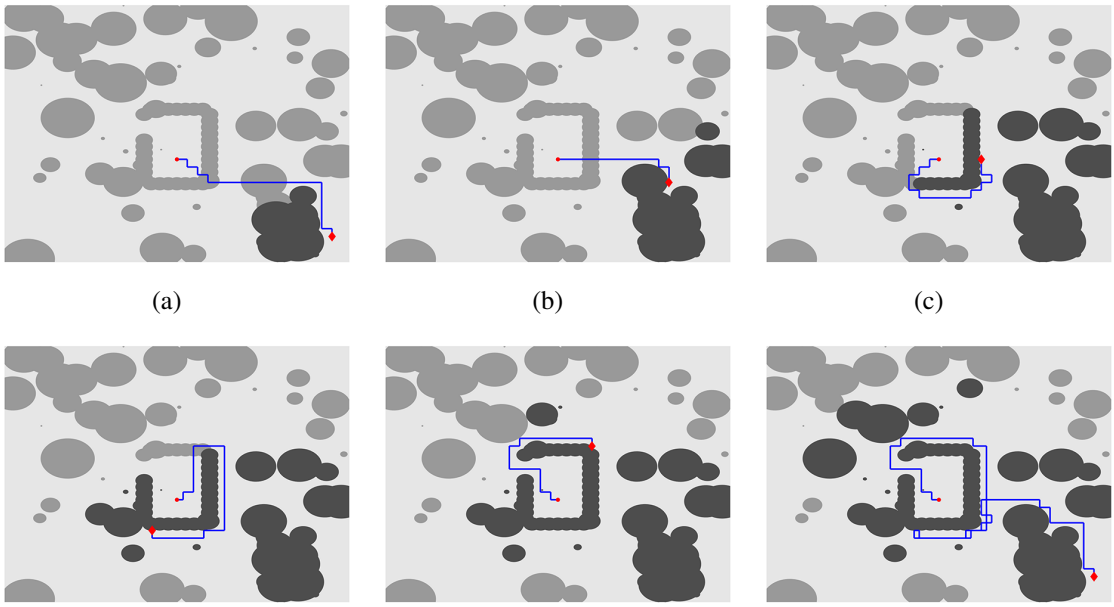


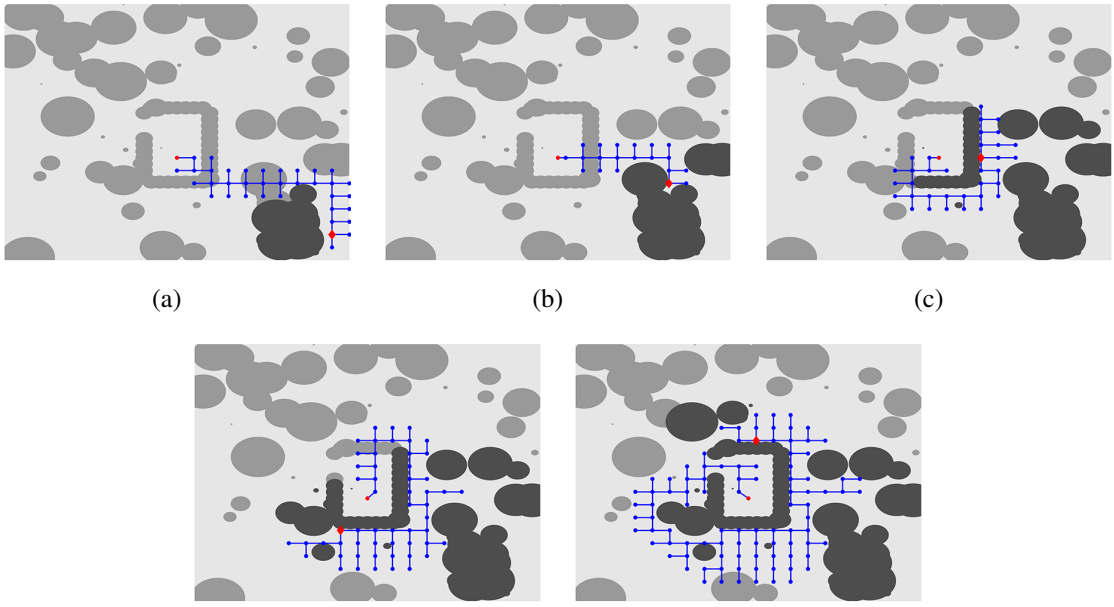
Figure 3.7: The graph produced under the same environment and target configuration as Figure 3.5 but with a different initial configuration located at top right corner. The graphs search almost the same region around the central box as those in Figure 3.5 except (a).



(a) (b) (c)

(d) (e) (f)

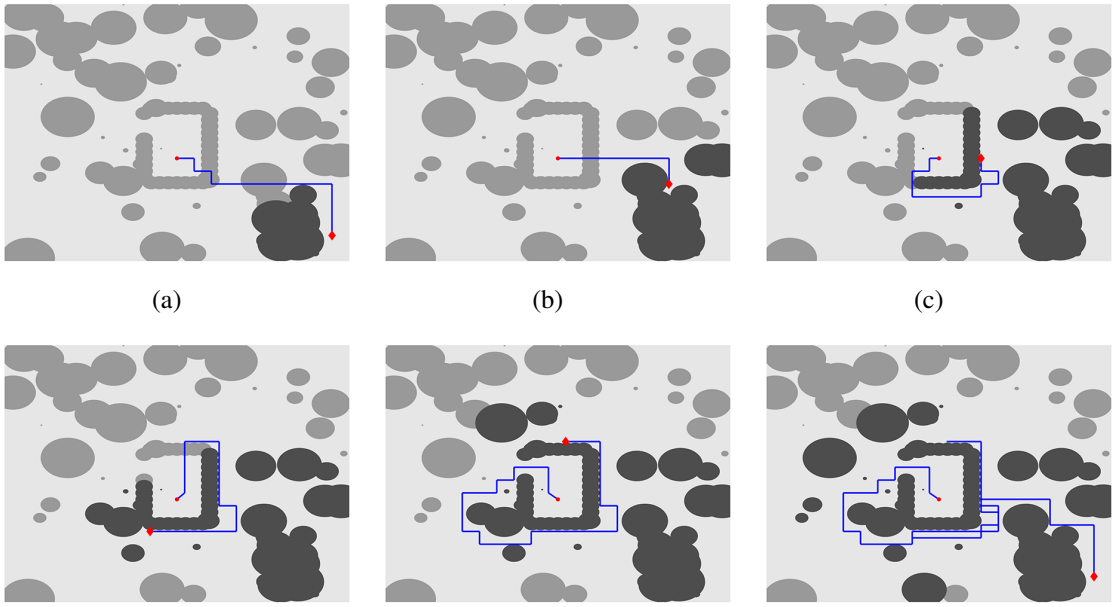
Figure 3.8: The paths calculated based on the results in Figure 3.7. (a)-(d) are middle steps that the robot stops because of the newly detected obstacles while moving and (e) is the path on which the robot get to the target. (f) gives the complete path of the robot. The paths are similar to those in Figure 3.6 in spite of different initial configuration.



(a) (b) (c)

(d) (e)

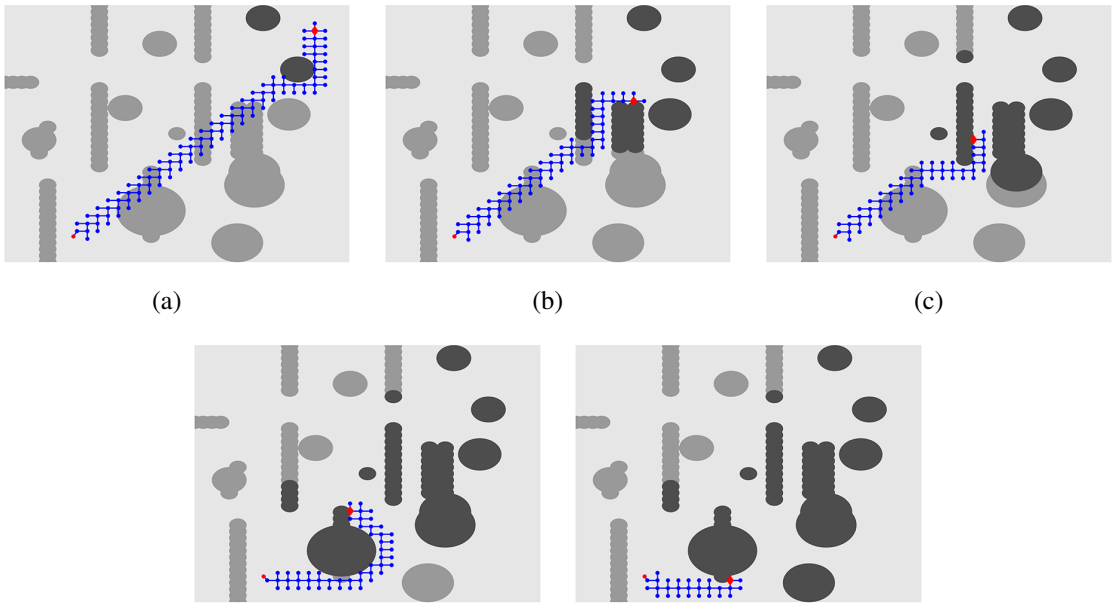
Figure 3.9: The graph produced under the same environment, initial and target configuration as Figure 3.7 but with a larger generating radius $l = 0.05$. The robot can no longer move through the original path, so to get to the target, it finds a different path that contains points with higher potential.



(a) (b) (c)

(d) (e) (f)

Figure 3.10: The paths calculated based on the results in Figure 3.9. (a)-(d) are middle steps that the robot stops because of the newly detected obstacles while moving and (e) is the path on which the robot get to the target. (f) gives the complete path of the robot.



(a) (b) (c)

(d) (e)

Figure 3.11: The generated graphs with $l = 0.03$ with one robot moving in a different environment. In that environment, a narrow corridor exists and the graph is able to get through from it.

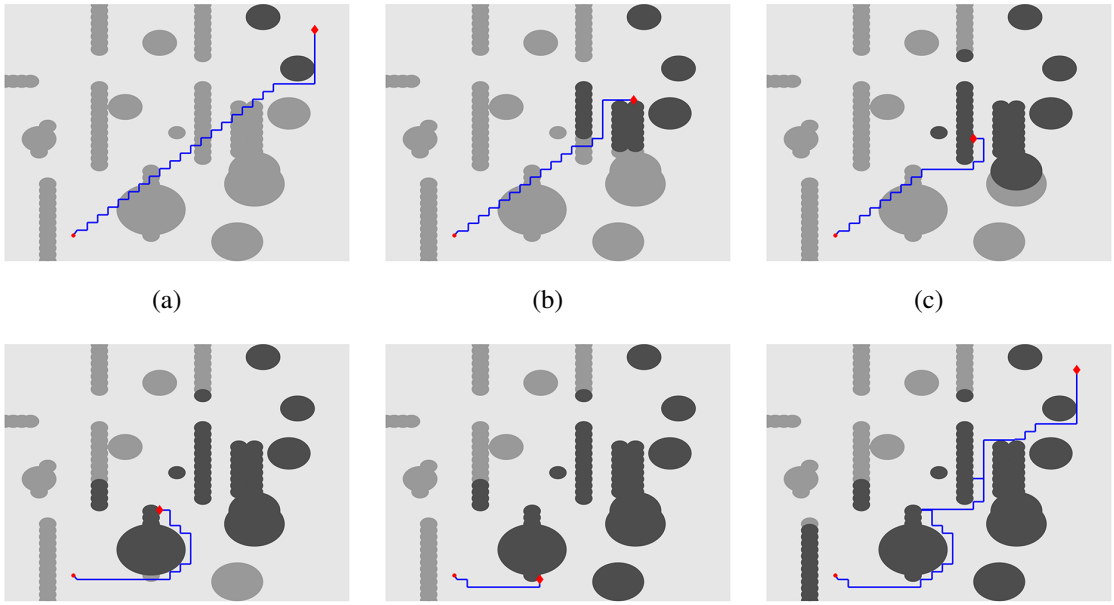


Figure 3.12: The paths calculated based on the results in Figure 3.11. (a)-(d) are middle steps that the robot stops because of the newly detected obstacles while moving and (e) is the path on which the robot get to the target. (f) gives the complete path of the robot.

Besides, the link between each pair of robots cannot be blocked by obstacles. All examples are accompanied by youtube videos, with the web links given in the footnotes. In Figure 3.13, a 2-robot (4 dimensional) system is used. From the pictures, we observe that the robots move up until trapped, because they always choose the fastest potential-decaying direction in the known environment. Then they retreat back and eventually find the correct way[1]. The next example is a 3-robot system (6 dimensional problem) shown in Figure 3.14. The environment allows a direct path from the initial to the target. The algorithm immediately finds a direct path and avoids taking any other sideways[2]. Finally, a 5-robot system is shown in Figure 3.15 to demonstrate that the algorithm is capable of solving a 10-dimensional problem with complicated environment[3], in which the robots need to twist so that they can successfully pass the obstacles. And an example of a 10-robot system (with 20 dimensional configuration space) moving in an unknown environment can also be found

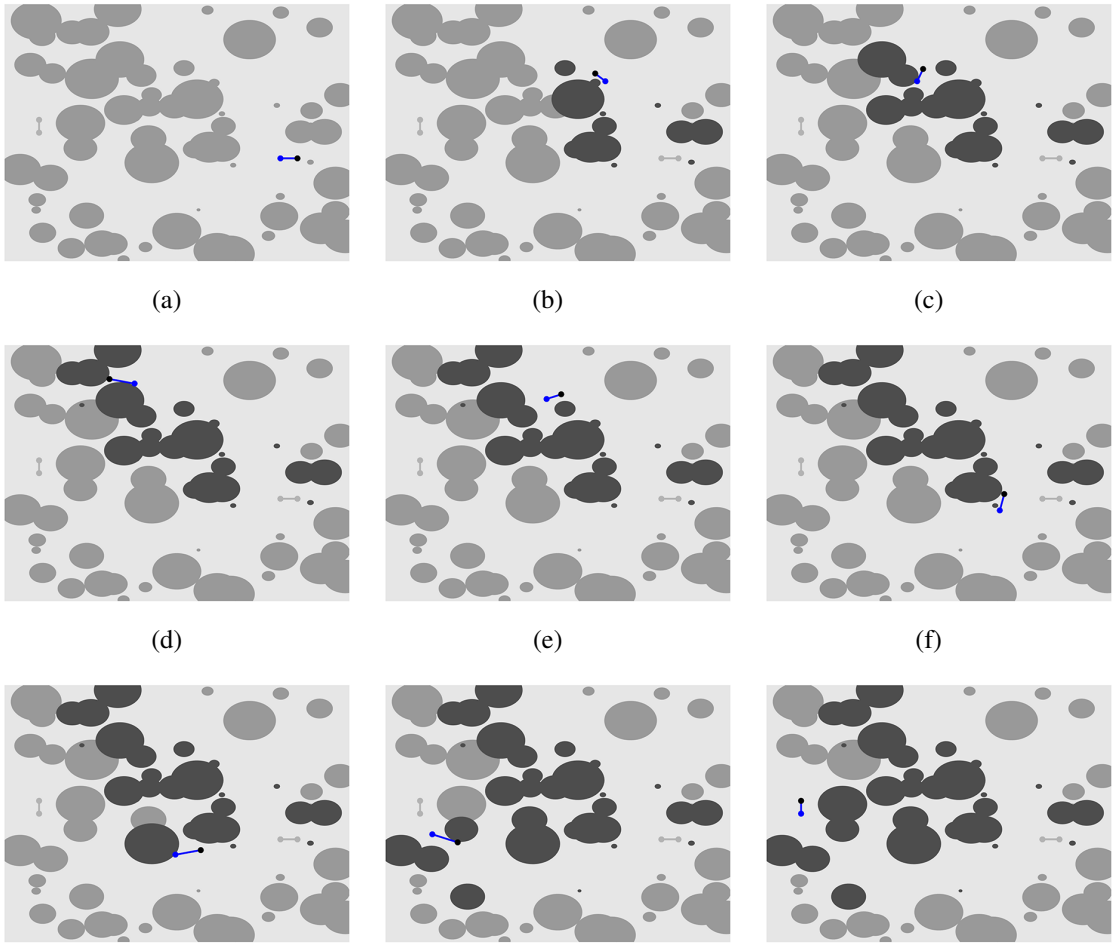
[1] video at https://youtu.be/6wKe7wnlG58
[2] video at https://youtu.be/q84VhKfYUyo
[3] video at https://youtu.be/H5lfzAYbfRA

online[4], which takes around 1 minute to do the whole calculation in Matlab on a Macbook Pro, the CPU of which is 2.9 GHz Intel Core i5, and no action is taken to optimize the performance of the algorithm.

In addition to the displayed paths, we illustrate the performance of the algorithms by using several other measurements. Table 3.1 shows the collective information about the number of vertices in graphs generated during the procedure. In this table, "Figure" column indicates the corresponding figures of the examples, "num of robots" represents the number of robots, and "l" is the generating radius in each experiment. To show the efficiency of our algorithms, we use the average number of nodes in the graphs, represented in "avg", and the maximum number of vertices amongst all graphs which is listed in the column "max". We can see that as the dimension of the problem (indicated in the "dim" column) increases, the size of the graphs increases, but not as fast as the exponential growth with respect to the dimensionality.

Furthermore, we observe that the algorithm generates the particular graph with the maximum number of vertices when the robots are trapped in local minimizer (shown in "trapped" column). In the 6 dimensional example, the robots do not encounter any local minimizer, which results in much fewer vertices. In fact, the sizes of graphs are smaller than those in the four-dimensional case. We also observe that the number of graphs generated by the algorithm ("num of G" column) highly relies on the environment and the choice of the orthonormal basis. Thus it is not used as a criterion to judge the efficiency of the algorithm. Overall, our algorithm is relatively efficient especially when dealing with high dimensional problems. The most costly part is to escape the local traps, and we propose a couple of strategies to improve the performance in the next section.

[4]video at https://youtu.be/gVinTsto7pE

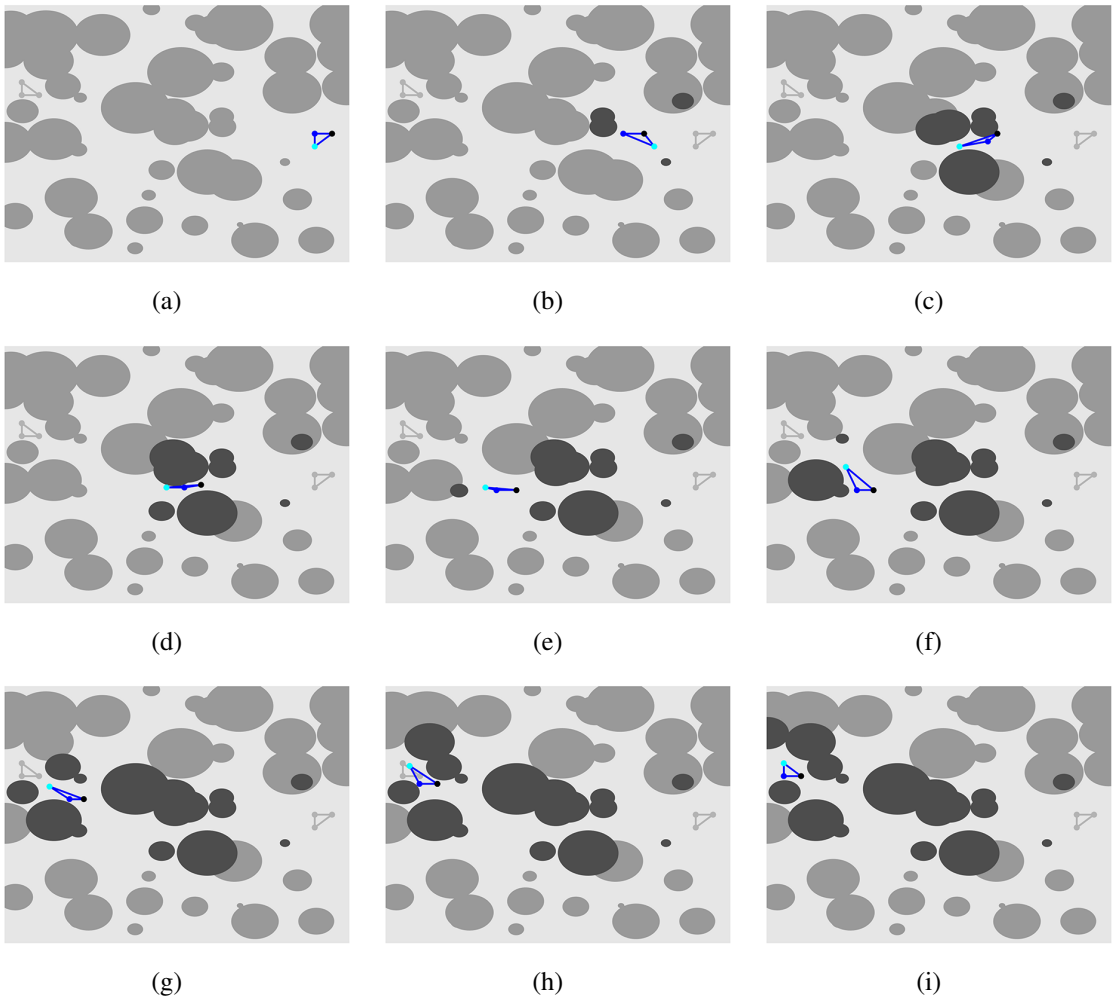


(a) (b) (c)

(d) (e) (f)

(g) (h) (i)

Figure 3.13: moving path for two robots with $l = 0.03$ with linking of each two robots not blocked by obstacles. Since the environment is unknown at first, the robots choose to move from the upper side without knowing it is a dead end. After recognizing they are trapped by obstacles, the robots move down to finally find a way to the target.

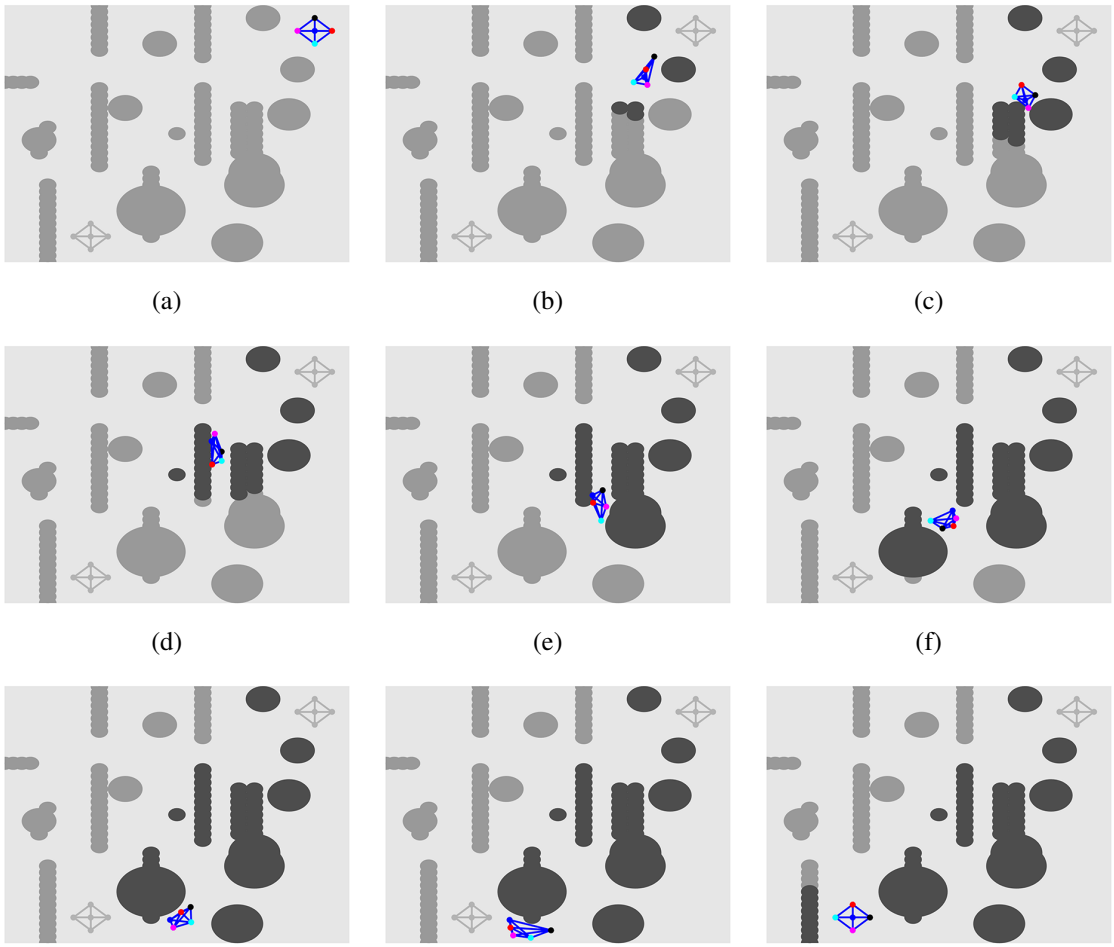


(a) (b) (c)

(d) (e) (f)

(g) (h) (i)

Figure 3.14: moving path for three robots with $l = 0.02$ with linking of each two robots not blocked by obstacles. There is a direct way to get to the target and the robots successfully find it without getting into traps because the algorithm is locally greedy.



(a) (b) (c)

(d) (e) (f)

(g) (h) (i)

Figure 3.15: moving path for five robots with $l = 0.03$ with linking of each two robots not blocked by obstacles. In this example, the robots change their shape to pass the narrow corridor and after getting into the local trap, they search around their way and move down to reach the destination.

Table 3.1: Information about number of vertices for the examples. This table shows that the number of nodes is increasing as the dimension of the problem increases but not as fast as exponential growth. And the number of nodes generated keeps small if the robots are not trapped in local minimum.

Figure	num of robots	l	dim	avg	max	trapped	num of G
Figure 3.5,3.6	1	0.03	2	60.5	150	yes	5
Figure 3.7,3.8	1	0.03	2	81.2	149	yes	5
Figure 3.9,3.10	1	0.05	2	47	94	yes	5
Figure 3.11,3.12	1	0.03	2	59.2	104	yes	5
Figure 3.13	2	0.03	4	606.7	3433	yes	9
Figure 3.14	3	0.02	6	632.4	1183	no	5
Figure 3.15	5	0.03	10	2178.4	6938	yes	7

Table 3.2: Information about number of vertices for the examples with escaping local traps algorithm. As we can see, the algorithm generates much fewer vertices compared to Table 3.1. And in the 5 robots system, the largest graph is no longer appears at local trap, instead the first generated graph is the with most nodes because of the physical distance from initial to target.

num of robots	l	dim	avg	max	num of G
2	0.03	4	212.4	295	8
5	0.03	10	1307	2492	7

3.4 Escaping Local Traps Rapidly

From the experiments conducted, we notice that the number of generated vertices increases when the robots are trapped in local minimizers, and the number of nodes at each local trap is proportional to the volume of the trap. This is not a surprise because the nearly exhausted search is used to escape local traps. In order to reduce cost, we present two different strategies. Before doing so, we need to identify local minimizers and define their trap regions. We say that a node point x is a local minimizer if no lower-potential points around x can be generated by Algorithm 3. Since a local trap can only be created by constraints because of the convexity of the potential function, we define the trap region as a set enclosed by the boundary of local constraints and the level curve (hyper-surface in high dimensional problems) of potential function in the following way:

$$\mathcal{L}^*(x) = \sup_c \{\mathcal{L}(c) : \mathcal{L}(a) \text{ is closed for all } a \in [c_0, c]\}, \tag{3.5}$$

where $c_0 = p(x)$ and $\mathcal{L}(c)$ is the closed set containing x with

$$\partial\mathcal{L}(c) \subset (\partial\mathcal{O}_c \cup \{x : p(x) = c\})$$

When a local trap is identified, our goal is to find points located on the intersection of obstacle boundary and the level curve (surface) given by (3.5) as quickly as possible, and then continue to generate vertices outside of the trap region. Here we introduce two different dimension reduction methods to achieve this goal.

3.4.1 Keep the robot near obstacles

: We know that some of the constraints in ϕ, ψ must be nearly activated around the local minimizer $x \in \partial\mathcal{O}$. For the ease of presentation, we denote those nearly activated constraints as $g_i(x) \leq \epsilon$ for some integer i where ϵ is a small positive number and g_i is some ϕ_j or ψ_k. For example, it can be chosen as $\epsilon = \min(\min_{k=1,2,\cdots,k_1} \phi_k(x), \min_{k=1,2,\cdots,k_2} \psi_k(x))$. We modify the algorithm so that it only generates points satisfying the inequalities, that is, only add points v such that $g_i(v) \leq \epsilon$ to V until there is a vertex $x \in \mathcal{E}$ with

$$\mathcal{E} = \left\{ x \in B(y, \sqrt{2}l) : \exists z \in \{c \in (x \pm \mathcal{N}) \backslash V : p(c) < p(x)\} \backslash \mathcal{O} \right\},$$

where $\mathcal{N}$ is the substituted set of orthonormal basis for N in the subspace, V is the current vertex set, $y = \arg\max_{g_i(z) \leq \epsilon} p(z)$ and g_i is the constraint mentioned above. After this point, we go back to Algorithm 3. With the same assumptions as stated in Theorems 3.2.1 and 3.2.2, we can show that this method find the path in finite steps, and the proof of the convergence follows the same arguments as provided in Section 3.6. Since $g_i(x) = \epsilon$ is continuous locally, the modified search is conducted in a low-dimensional subspace if ϵ is chosen appropriately.

3.4.2 Fix the shape formed by robots

: A different way to get out of the local traps is to introduce a set of new constraints $\{h_1(x) = 0, \cdots, h_k(x) = 0\}$ on the robots so that they restrict the graph generation in a low dimensional subspace $\hat{\Omega}$. For example, one may fix the pairwise distance between robots, so $h_i(x) = 0$ indicates that the distance between a certain pair of robots is a given value. In 2-D or 3-D workspace, those restrictions often lead to a fixed shape formed by the robots. Each h_i reduces the search dimension by one because the new vertices added to V must satisfy $h_i(x) = 0$. Similar to the previous strategy, we stop this procedure when a vertex $x \in \mathcal{E}$ is generated, which indicates the robots moved out of the known local trap. On the other hand, it is possible that after adding new constraints, there is no feasible way to move out. In this case, no new vertex can be generated in V, then we remove one of the added constraints, and continue with the graph generating algorithm in a subspace which is one dimension higher than the previous subspace. The procedure is repeated if necessary. For this method, if we further assume that there is a feasible tube in the low dimensional subspace defined by all constraints, including the added ones $h_i = 0$, we can use the same proof to show its convergence in a finite number of steps. In this chapter, we implement this method in our high dimensional examples.

In the two and five robots cases demonstrated in Section 3.3, we fix the distance between each pair of robots when a local minimizer is encountered. To compare the results, we carry out several new experiments, in which all set-ups including initial and target configurations, the obstacles and all parameters are the same. The final path and how the robots move can be found in videos[5]. The information on the generated graphs is displayed in Table 3.2, where we can see that the number of vertices decreases significantly. In the 5 robots case, the largest graph is no longer produced at local traps. Instead, the first generated graph contains more nodes because of the long distance from initial to target configurations. In

[5] video for two robots with the improved algorithm at https://youtu.be/od5fmuo8cR8 and video for five at https://youtu.be/vVHThxmtmf8

our examples, we observe that nodes needed around the local minimizers are reduced from $O(\alpha^n)$ to $O(\alpha^2)$, where α is the edge length assuming the local trap is a square and n is the dimension of Ω. In addition, we would like to mention that it usually takes less than a minute to compute the path for a system with 20 robots using MATLAB on a typical laptop computer (MacBook with 2.7GHz Intel Core i5 processor).

We also observe a common feature in all examples: the environment is not entirely explored, and the generated graphs are greedily expanding towards the target configuration. This special feature is not by accident. In fact, it is determined by the Fokker-Planck equation in optimal transport theory. We give a thorough discussion on their connections in the next section.

3.5 Relation to FPE and Optimal Transport

The design of the graph generating algorithm, Algorithm 3, is inspired by the evolution of FPE, which determines a region $\mathcal{R}_f$ where the search is conducted. In this section, we describe in detail on how the region evolves following the solution of FPE,

$$\begin{cases} \frac{\partial \rho}{\partial t}(x,t) = \nabla \cdot (\rho(x,t)\nabla p(x)) + \beta \Delta \rho(x,t) \\ \rho(x,0) = \rho_0(x) \end{cases}, \tag{3.6}$$

where ρ_0 is a given distribution and $p(x)$ is the potential function. Based on (3.5), the region $\mathcal{R}_f$ is constructed by an intermittent diffusion process, meaning we take β to be 0, so that the density is transported greedily along the negative gradient direction, while we adjust $\beta > 0$ to trigger a diffusion process when trapped in a local minimizer. For simplicity, we call $\beta = 0$ the gradient part and $\beta > 0$ the diffusion part. However, since our graph generating algorithm only choose new points along the given orthonormal directions N,

we must replace $\nabla p(x)$ in (3.6) by its projection onto N:

$$\mathbf{u}(x) = \sum_{y \in \mathcal{H}(x)} P_y \nabla p(x),$$

where P_y is the projection operator to y, and $\mathcal{H}(x)$ is defined as,

$$\mathcal{H}(x) = \arg\min_{y \in \mathcal{A}} \frac{\langle \nabla p(x), y \rangle}{\|\nabla p(x)\|}$$

with $\mathcal{A} = \{y : y \in N \text{ or } -y \in N \text{ and } x + \lambda y \notin \mathcal{O} \forall \lambda \in (0, \xi) \text{ for some } \xi > 0\}$. The resulting equation is

$$\begin{cases} \frac{\partial \rho}{\partial t}(x,t) = \nabla \cdot (\rho(x,t)\mathbf{u}(x)) + \beta \Delta \rho(x,t) \\ \rho(x,0) = \rho_0(x) \end{cases} . \tag{3.7}$$

We note that both (3.6) and (3.7) can be rewritten as

$$\frac{\partial \rho}{\partial t}(x,t) = \nabla \cdot \{\rho(x,t)\left[\mathbf{v}(x) + \nabla(\beta \log \rho(x,t))\right]\},$$

where $\mathbf{v} = \nabla p$ for (3.6) and $\mathbf{v} = \mathbf{u}$ for (3.7). This expression can be approximated by the following upwind discretization of (3.6) on a lattice grid $GL \subset \Omega \backslash \mathcal{O}$ (here we assume that x_i is one of the grid points), with mesh size Δx and orientation aligned with the orthonormal basis N used in Algorithm 3 [52],

$$\begin{aligned} \frac{\partial \rho_j}{\partial t} = \Big(& \sum_{k \in Nb(j)} (F_k(\rho,\beta) - F_j(\rho,\beta))_+ \rho_k d_{jk} \\ & - \sum_{k \in Nb(j)} (F_j(\rho,\beta) - F_k(\rho,\beta))_+ \rho_j d_{jk} \Big) \frac{1}{(\Delta x)^2}, \end{aligned} \tag{3.8}$$

where $(\cdot)_+ = \max(\cdot, 0)$, $\rho_j = \rho(x_j, t)$, $Nb(j)$ is the set of all adjacent nodes (neighbors) of node x_j on the grid GL, $F_j(\rho, \beta) = \frac{\partial}{\partial \rho_j}\mathcal{F}(\rho, \beta)$, in which $\mathcal{F}(\rho, \beta)$ is the free energy

$$\mathcal{F}(\rho, \beta) = \sum_{j=1}^{N} \left(p(x_j)\rho_j + \beta\rho_j \log \rho_j\right), \tag{3.9}$$

and $d_{jk} = d_{kj} = 1$ for (3.6). A similar discretization can be derived for (3.7). The value of d_{jk} in the discretization of $\mathbf{u}$, which, if assume $\phi(j) > \phi(k)$ without loss of generality, can be defined as

$$d_{jk} = \begin{cases} 1 & \langle j-k, \nabla p(j)\rangle = \min_{\{i \in Nb(j): p(i) < p(j)\}} \langle j-i, \nabla p(j)\rangle \\ 0 & \text{otherwise} \end{cases}, \tag{3.10}$$

where i, j, k represent the coordinates of the corresponding nodes and $\nabla p(j)$ is the gradient vector at configuration j in Ω. And if the projection is not involved as is in the diffusion part, we simply let $d_{jk} = 1$ for all j, k.

In the rest of this section, we show how to build $\mathcal{R}_f$ using (3.8) with $\Delta x = l$. The strategy is that we alternate the procedures between the gradient ($\beta = 0$) and diffusion ($\beta \neq 0$) to grow the region. When a new part of region is formed each time, we simply union it with the existing one. We want to mention that at any point we change the procedures (β from 0 to a nonzero value, or vice versa), we reinitialize the density before evolving (3.7). We terminate the procedure, if the target configuration is included in $\mathcal{R}_f$.

3.5.1 Gradient part of $\mathcal{R}_f$

For the gradient case, the points on the grid expand along the projection of the negative gradient of the potential function onto N. We evolve (3.8) with $\beta = 0$ and the initial condition

$$\rho(x, 0) = \delta_{x_i} = \begin{cases} 1 & x = x_i \\ 0 & x \neq x_i \end{cases},$$

where x_i is the starting point of the current gradient procedure. Until reaching the steady state, the solution $\rho(x,t)$ on the grid GL can be calculated. The steady state solution satisfies for the following property,

Proposition 2. *$\rho(x,\infty) = \delta_{V_{loc}}$, where $V_{loc} \subset GL$ is a subset of local minimizers of $p(x)$ on the given grid.*

Then we select points such that

$$\mathcal{R}_1(x_i) = \left\{ x \in GL : \frac{\partial}{\partial t}\rho(x,t) > 0 \text{ for some } t > 0 \right\}, \tag{3.11}$$

Once $\mathcal{R}_1(x_i)$ is determined, we merge it to the set $\mathcal{R}$ constructed in the previous steps (we use empty at the first step), i.e. $\mathcal{R} = \mathcal{R} \bigcup \mathcal{R}_1$. If $x_i \neq x_f$, we continue to amend the set $\mathcal{R}$ with the diffusion procedure described in Section 3.5.2.

3.5.2 Diffusion part of $\mathcal{R}_f$

We assume that the previously constructed set is $\mathcal{R}_p = \mathcal{R}$. In the diffusion part, since $\beta > 0$, $\log \rho$ is involved in the calculation. To avoid blowing up in the computation, we initialize the density β for (3.8) as below:

$$\rho(x,0) = \delta_{\mathcal{R}_p} = \begin{cases} \frac{1}{|\mathcal{R}_p|}(1-\epsilon) & x \in \mathcal{R}_p \\ \frac{1}{|GL \backslash \mathcal{R}_p|}\epsilon & x \in GL \backslash \mathcal{R}_p \end{cases},$$

where ϵ can be an arbitrarily small positive real number. With this initialization, we can calculate $\rho(x,t)$ in (3.8) until reaching the stationary solution $\rho(x,\infty)$. Now following

$\rho(x, \infty)$, we choose points on the grid as:

$$\begin{aligned}
\mathcal{R}_2 &= \bigcup_{s=0}^{W} \mathcal{R}_2^s, \\
\mathcal{R}_2^s &= \left\{ x = \arg \max_{y \in GL \setminus \bigcup_{j=0}^{s-1} \mathcal{R}_2^j} \rho(y, \infty) : \mathcal{R}_2^{s-1} \bigcap Nb(x) \neq \emptyset \right\}, \\
\mathcal{R}_2^0 &= \mathcal{R}_p,
\end{aligned}$$

where W is defined by

$$W = \arg\min_s \left\{ s : z \in \mathcal{R}_2^s, \exists y \in Nb(z) \setminus \left(\bigcup_{\tau=1}^{s} \mathcal{R}_2^\tau \right) \text{ with } p(y) < p(z) \right\}.$$

We union $\mathcal{R}_2$ into $\mathcal{R}$ by defining $\mathcal{R} = \mathcal{R} \bigcup \mathcal{R}_2$. In the newly selected $\mathcal{R}_2$, we pick

$$x_i = \arg\min_y \{p(y) : y \in Nb(x) \setminus \mathcal{R} \text{ for some } x \in \mathcal{R}\}. \tag{3.12}$$

This x_i is the new starting point for the next gradient procedure, and we return to the gradient part as described in Section 3.5.1.

By alternating the procedures to obtain $\mathcal{R}_1$ and $\mathcal{R}_2$ until x_f is included, we define the final region

$$\mathcal{R}_f = \bigcup_{x \in \mathcal{R}} Box(x, l)$$

where $Box(x, l)$ is the closed box centered at x with edge length $2l$. With $\mathcal{R}_f$, we have the following theorem.

Theorem 3.5.1. *Assuming that the robots only stops on node points with assumptions in Theorem 3.2.2 and $R > L$, the complete path γ generated by the algorithm satisfies $\gamma \subset \mathcal{R}_f$.*

We show two examples in Figure 3.16 with initial configurations indicated by red diamonds and target red circles. The gray region is calculated by the gradient and diffusion

procedures described in this section. The computation is done on a grid with mesh size $\Delta x = l$. As we can see clearly in Figure 3.16(a), starting from the right middle part of Ω, the graph G first expands along the x-axis, which is the projected negative gradient direction on the lattice grid, until it hits an obstacle. Then the procedure is switched to the diffusion case, and it produces a region in front of the obstacle following the Gibbs' distribution until it finds a way out. After that, it changes back to the gradient case and moves to the target greedily. This time, the projected negative gradient direction coincides with the actual one. Again, the diffusion case kicks in when a local minimizer is encountered. The procedure repeats until the target is reached. Figure 3.16(b) shows another example with more complicated set ups.

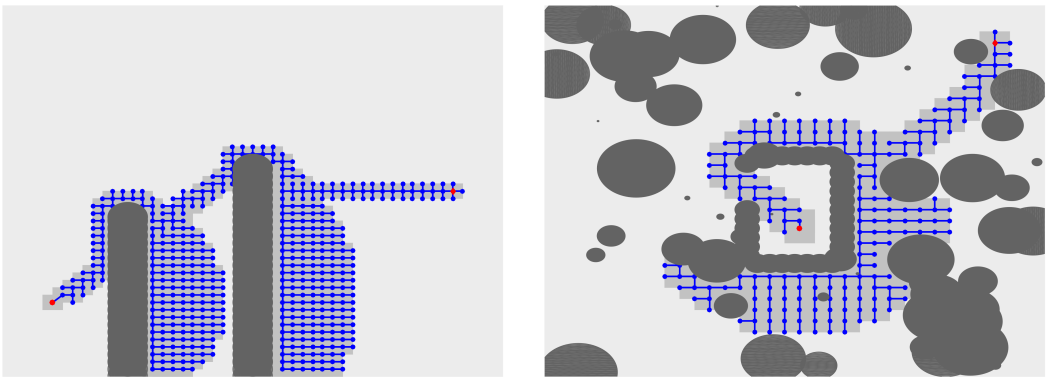
We would like to mention that we use the forward Euler method to discretize (3.8) in time,

$$\begin{aligned}\rho_j^+ = \rho_j + \Big(&\sum_{k\in Nb(j)} (F_k(\rho,\beta) - F_j(\rho,\beta))_+ \rho_k d_{jk} \\ &- \sum_{k\in Nb(j)} (F_j(\rho,\beta) - F_k(\rho,\beta))_+ \rho_j d_{jk} \Big) \frac{\Delta t}{(\Delta x)^2}.\end{aligned} \tag{3.13}$$

To make the scheme convergent, we need the following proposition, which can be regarded as the Courant-Friedrichs-Lewy (CFL) conditions for numerical PDE schemes.

Proposition 3. *Given the lattice grid with grid size* Δx*,* (3.13) *is stable if*

$$\begin{aligned}\Delta t < \min \Big\{ &\frac{1}{\max_{j\in GL} \sum_{k\in Nb(j)} (F_j - F_k)_+ d_{jk}}, \\ &\min_{j\in GL} \frac{1-\rho_j}{\sum_{k\in Nb(j)} (F_k - F_j)_+ \rho_k d_{jk}} \Big\} (\Delta x)^2,\end{aligned} \tag{3.14}$$



(a) (b)

Figure 3.16: Relation between graph generator and FPE: Given an known environment, the graph generated by Algorithm 3 can be fully covered by $\mathcal{R}_f$. In both 3.16(a) and 3.16(b), there exists gradient and diffusion part. The gray shadow is $\mathcal{R}_f$ and the blue part is the graph G with initial and target being marked as red diamond and red dot respectively.

Proof. To make the scheme stable, we need

$$\begin{cases} \rho_j + \sum_{k\in Nb(j)}(F_k - F_j)_+\rho_k d_{jk}\frac{\Delta t}{(\Delta x)^2} < 1 \\ \rho_j - \sum_{k\in Nb(j)}(F_j - F_k)_+\rho_j d_{jk}\frac{\Delta t}{(\Delta x)^2} > 0 \end{cases}$$
$$\Longrightarrow \begin{cases} \Delta t < \frac{1}{\sum_{k\in Nb(j)}(F_j-F_k)_+ d_{jk}}(\Delta x)^2 \\ \Delta t < \frac{1-\rho_j}{\sum_{k\in Nb(j)}(F_k-F_j)_+\rho_k d_{jk}}(\Delta x)^2 \end{cases}$$

for all $j \in GL$. This leads to (3.14). In addition, following the proof of Theorem 3 in [53], we can obtain $\rho_i(t) \geq \epsilon$ for a fixed grid. Therefore $(F_j - F_k)_+$ and $(F_j - F_k)_+\rho_k$ are bounded from above for all edges $\{(j,k)\}$ in GL. This implies that the right hand side of (3.14) is bounded from below by a positive real number, so Δt can stay strictly positive. □

Remark 2. *For each given l, we can get a region $\mathcal{R}_f(l)$. If we let l go to 0, which means that the lattice GL approaches to the continuous space, we can define $\mathcal{R}_f(\infty) = \limsup_{l\to 0}\mathcal{R}_f(l)$. The graph G produced by our algorithm must satisfy $G \subset \mathcal{R}_f(\infty)$. In fact, $\mathcal{R}_f(\infty)$ is the smallest bounded region in which the search is conducted.*

3.6 Convergence Analysis

3.6.1 Convergence of the algorithm

In this section, we give detailed proofs for the Theorems 3.2.1 and 3.2.2. To do this, we need to prove several lemmas first. We begin by showing that the set of all feasible path Γ is compact in a finite time interval $[0, T]$ (Lemma 3.6.1) and there exists a feasible path within a tubular region $\mathcal{T}$ that is clear of obstacles (Lemma 3.6.2).

Lemma 3.6.1. *If there exists a feasible path, Γ is non-empty and compact with respect to the L^∞ norm given by*

$$d_\Gamma(\gamma_1, \gamma_2) = \max_{t\in[0,T]} \|\gamma_1(t) - \gamma_2(t)\|,$$

where γ_1 and γ_2 are two paths in Γ.

Proof. Let us denote the feasible path as

$$\gamma : [0, T_0] \to \Omega.$$

If we define $\gamma(t) = x_t$ for all $t \in (T_0, T]$, we have $\gamma \in \Gamma$. To prove Γ is compact, we assume there is a sequence of paths $\{\gamma_i\}_{i=0}^{\infty} \subset \Gamma$ such that

$$\lim_{i\to\infty} d_\Gamma(\gamma_i, \gamma) = 0.$$

Since Ω is compact and $\mathcal{O}$ is open, it implies that $(\Omega\backslash\mathcal{O})$ is compact. Therefore for any

$t \in [0,T]$, $\gamma_i(t) \to \gamma(t)$ as $i \to \infty$, we have $\gamma(t) \in (\Omega \backslash \mathcal{O})$. This includes

$$\gamma(0) = \lim_{i\to\infty} \gamma_i(0) = x_s,$$
$$\gamma(T) = \lim_{i\to\infty} \gamma_i(T) = x_t,$$
$$\gamma(t) \subset (\Omega \backslash \mathcal{O}),$$

which gives $\gamma \in \Gamma$, and Γ is a close set. In addition, since Ω is compact, Γ is bounded, we conclude that Γ is compact. □

Lemma 3.6.2. *Assume that (3.2) is true, there must exist a feasible path $\gamma \in \Gamma$ satisfying*

$$\left(\cup_{t\in[0,T]} B(\gamma(t), L)\right) \cap \mathcal{O} = \emptyset,$$

where $\mathcal{O}$ is the constrained set.

Proof. Because of (3.2), there exists a sequence of paths $\{\gamma_n\} \subset \Gamma$ satisfying

$$\lim_{n\to\infty} \inf_{t\in[0,T]} \sup_{r\in[0,\infty)} \{r : B(\gamma_n(t), r) \cap \mathcal{O} = \emptyset\} = L.$$

From Lemma 3.6.1, we know that Γ is compact, therefore $\lim_{n\to\infty} \gamma_n = \gamma_0 \in \Gamma$. For an arbitrary $t \in [0,T]$, denote

$$L(t) = \sup_{r\in[0,\infty)} \{r : B(\gamma_0(t), r) \cap \mathcal{O} = \emptyset\}$$

As the whole space Ω is compact, the limit $L(t) < \infty$. Since $\inf_{t\in[0,T]} L(t) = L$, one has $L \leq L(t)$ for arbitrary time $t \in [0,T]$. That is, fix the curve γ_0 defined as above, for all $t \in [0,T]$, $B(\gamma_0(t), L) \cap \mathcal{O} = \emptyset$ and thus

$$\left(\cup_{t\in[0,T]} B(\gamma_0(t), L)\right) \cap \mathcal{O} = \emptyset,$$

which proves the lemma. □

In the next few lemmas, we prove several results that ensure the generating graph algorithm (Algorithm 3) creating new points in the feasible region when the radius l is small enough compared to L, and the process does not stop until reaching a neighborhood of the target configuration.

Lemma 3.6.3. *Given a point x on an n-dimensional Euclidean space and $L > 0$. Let $y \in S(x, L)$, and N be a set containing n orthonormal vectors, then $\exists z \in K = \{y \pm lN : y \pm le, e \in N\}$ such that $z \in B(x, L)$ if*

$$0 < l < \frac{2L}{\sqrt{n}}.$$

Proof. Without loss of generality, we can assume $x = (0, \cdots, 0)$. Denote $y = (y_1, \cdots, y_n)$, we can rewrite $K = \{z_k = (y_1, \cdots, y_k \pm l, \cdots, y_n)\}_{k=1}^n$. Since $y \in S(x, L)$, then $\exists k \in \{1, \cdots, n\}$, so that $|y_k| \geq \frac{L}{\sqrt{n}}$. Consider the point $z_k = (y_1, \cdots, y_k - sign(y_k)l, \cdots, y_n)$, then

$$\begin{aligned}
\|z_k - x\|^2 &= \sum_{i \neq k} y_i^2 + (y_k - sign(y_k)l)^2 \\
&= L^2 + l^2 - 2sign(y_k)y_k l \\
&= L^2 + l^2 - |y_k|l \\
&\leq L^2 + l^2 - \frac{2L}{\sqrt{n}}l.
\end{aligned}$$

One has $L^2 + l^2 - \frac{2L}{\sqrt{n}}l < L^2$ if $0 < l < \frac{2L}{\sqrt{n}}$. So $z_k \in B(x, L)$. □

Lemma 3.6.4. *Given a continuous path γ and a closed set $\mathcal{V} \subset \Omega$ with $\gamma(0) = x, \gamma(T) = y$, and $x \in \mathcal{V}$ and $y \notin \mathcal{V}$, then there exists $z \in \gamma$ such that $z \in \partial\mathcal{V}$ and $\gamma \cap \partial\mathcal{V}$ is closed.*

Proof. We use the signed distance function $f(u, \mathcal{V})$ with $f > 0$ if $u \in \mathcal{V}$ and $f < 0$ when

$u \notin \mathcal{V}$. It is clear that f is continuous with respect to u. Hence,

$$g = f \circ \gamma : [0, T] \to \Omega$$

is also continuous with $g(0) \geq 0$ and $g(T) \leq 0$ since $\gamma(T) = y \notin \mathcal{V}$ and $\gamma(0) = x \in \mathcal{V}$. As a result, there is at least one point $t_1 \in [0, T]$ so that $g(t_1) = 0$, hence $\gamma(t_1) \in \partial\mathcal{V}$. In fact, all points satisfying $f(u) = 0$ are on $\partial\mathcal{V}$.

Assume $\gamma \cap \partial\mathcal{V}$ is open, $\mathcal{B} = (f \circ \gamma)^{-1}(\gamma \cap \partial\mathcal{V})$ must be open, because $(f \circ \gamma)$ is continuous. This implies that $\mathcal{B}$ is the disjoint union of some open intervals. Take one of the open interval, say (a, b), we have $a \notin \mathcal{B}$ and $f \circ \gamma((a, b)) = 0$. Due to the continuity of $f \circ \gamma$, we have $f \circ \gamma(a) = 0$, which means $a \in \mathcal{B}$, and this is a contradiction. Therefore, $\gamma \cap \partial\mathcal{V}$ must be closed. □

Lemma 3.6.5. *Assume that (3.2) holds. Graph $G = (V, E, K, p)$ is generated by Algorithm 3 with $l \leq \frac{2L}{\sqrt{n}}$. If $x_t \notin V$, then the graph generating step of Algorithm 3 does not stop, and there exists at least one point in the feasible region that can be added to G by the algorithm.*

Proof. Let us assume that the graph generating algorithm terminates after finite steps, returning a connected graph $G = (V, E)$ containing x_s. We denote

$$C = \cup_{v \in V} \bar{B}(v, L),$$

which is a closed set with $x_s \in C$. First we want to prove $x_t \in C$ by contradiction. Let us assume $x_t \notin C$. Take the path γ in Lemma 3.6.2, it is true that $\gamma(0) = x_s \in C$ while $\gamma(T_0) = x_t \notin C$. Since γ is continuous, there exists at least one point that γ intersects with ∂C by Lemma 3.6.4, and we denote it as $\{\gamma(t_i)\}$. Let $x = \sup_{t_i} \gamma(t_i)$ be the last intersection point along the path. By Algorithm 3, we can find a vertex $v_c \in V$ such that $x \in S(v_c, L)$, which is the sphere centered at v_c with radius L. Since $x \in \gamma$ and by (3.2), we know that $B(x, L) \subset (\Omega \backslash \mathcal{O})$. Further we claim that there is no $v \in V \cap B(x, L)$,

otherwise, $x \in B(v, L)$ implies $x \in C^o$, which is a contradiction with $x \in \partial C$.

Since the algorithm stopped, all current vertices $v \in V$ must have been tried to generate points around them. When the vertex v_c is chosen, by Lemma 3.6.3, at least one point p can be generated in $B(x, L)$, which means either the algorithm should not terminate without having p, or p already exists before, which contradicts there is no vertex of G in $B(x, L)$. Therefore, we conclude that $x_t \in C$ if the algorithm stops.

If $x_t \in C$, since $x_t = \gamma(T_0)$, $B(x_t, L) \subset (\Omega \backslash \mathcal{O})$, then there is at least one vertex $v \in B(x_t, L)$, by the algorithm, an edge between x_t and v should be added to the graph G. Thus, if the algorithm stops, x_t and x_s are in the same connected component in the graph G. □

Now, we are ready to prove the main theorems.

Proof. (Proof of Theorem 3.2.1) Since Ω is compact and the graph G is a subset of grid points with length l in Ω, G must contain a finite number of vertices. Otherwise, there is a cluster point in the vertex set, which implies that there exist two vertices in G whose distance is strictly smaller than the generating radius l regardless how small l is. This contradicts to the fact that that G is a subset of a grid with the smallest distance between any two points is l. From the construction mechanism, G is always connected. Then Lemma 3.6.5 implies that G connects x_s and x_t. □

Proof. (Proof of Theorem 3.2.2) We denote the detected constraints after step i as $\mathcal{O}_c^i$. If the algorithm does not stop in finite steps, it means $m = \infty$. Because Ω is compact and $\mathcal{O}_c^i$ is open for every i, then $\Omega \backslash \mathcal{O}_c^i$ is also compact. Thus, there exists a cluster point for the stopping point set $\{\gamma_k(T_k)\}_{k=1}^{\infty}$. For an arbitrary $\epsilon > 0$, i can be chosen so that $\|\gamma_i(T_i) - \gamma_j(T_j)\| < \epsilon$ for all $j > i$. Choose $\epsilon < R - q$. Without loss of generality, one can assume that γ_i is obtained before γ_j, and at j-th step, the algorithm stops because $\mathcal{O}_c \cap \gamma_j \neq \emptyset$. Even more,

$$\bar{B}(\gamma_i(T_i), R) \cap \mathcal{O}_c \cap \gamma_j \neq \emptyset.$$

And since

$$\bar{B}(\gamma_i(T_i), R) \cap \mathcal{O}_c = \mathcal{O}_c^i,$$

it is true that

$$\mathcal{O}_c^i \cap \gamma_j \neq \emptyset$$

which is a contradiction since for each generated graph $G = (V, E, p)$ at step j,

$$V \cap \mathcal{O}_c^i = \emptyset \text{ and } E \cap \mathcal{O}_c^i = \emptyset$$

for all $i < j$, which concludes $m < \infty$. □

3.6.2 Proof of the bounded searching region

In this section, we prove Theorem 3.5.1 and related propositions. First, we give a detailed proof for the following lemma.

Lemma 3.6.6. *Given $\beta \geq 0$,* (3.8) *is convergent with $\mathcal{F}(\rho, \beta)$, given in* (3.9)*, being a Lyapunov function. When $\beta \neq 0$,* (3.8) *converges, with any given initial condition, to the Gibbs' distribution*

$$\rho(x) = \frac{1}{K} \exp\left(-\frac{p(x)}{\beta}\right),$$

where K is the normalization constant making ρ a density function.

Proof. Taking the derivative along the solution $\rho(x,t)$ of (3.8), we have

$$\begin{aligned}
&\frac{d}{dt}\mathcal{F}(\rho(t)) = \sum_{j=1}^{|V|} F_j \frac{d\rho_j}{dt} \\
=&\sum_{j=1}^{|V|}\sum_{k\in Nb(j)} (F_k(\rho,\beta) - F_j(\rho,\beta))_+ \rho_k d_{jk} F_j - \sum_{j=1}^{|V|}\sum_{k\in Nb(j)} (F_j(\rho,\beta) - F_k(\rho,\beta))_+ \rho_j d_{jk} F_j \\
=&\sum_{j=1}^{|V|}\sum_{k\in Nb(j)} (F_j(\rho,\beta) - F_k(\rho,\beta))_+ \rho_k d_{jk} F_k - \sum_{j=1}^{|V|}\sum_{k\in Nb(j)} (F_j(\rho,\beta) - F_k(\rho,\beta))_+ \rho_j d_{jk} F_j \\
=&-\sum_{j=1}^{|V|}\sum_{k\in Nb(j)} (F_j(\rho,\beta) - F_k(\rho,\beta))_+^2 \rho_k d_{jk} \leq 0
\end{aligned}$$

Because $\mathcal{F}(\rho,\beta)$ is bounded from below, (3.8) is convergent. One can check directly that the Gibbs' distribution is the stationary solution of (3.8), and it is also the minimizer of $\mathcal{F}(\rho,\beta)$. □

Proof. (Proof of Proposition 2) First of all, by Lemma 3.6.6, (3.8) converges when $\beta = 0$. We assume that the support of one stationary solution $\rho(x)$ contains a point other than local minimizers, say $x_0 \in GL$. Then there exists $z \in Nb(x_0)$ with $p(z) < p(x_0)$. By the definition of d_{ij}, there must be an edge linking x_0 and z such that $d_{x_0 z} = 1$ and $p(z) < p(x_0)$. We construct

$$\hat{\rho}(x) = \begin{cases} \rho(x) & x \neq x_0, x \neq z \\ 0 & x = x_0 \\ \rho(z) + \rho(x_0) & x = z \end{cases}$$

It is easy to check $\mathcal{F}(\hat{\rho}) < \mathcal{F}(\rho)$, which leads to a contradiction that ρ is the minimizer of $\mathcal{F}(\rho)$. □

Given an arbitrary environment $\mathcal{O}$, let us denote $G_f = (V_f, E_f)$ the graph generated by Algorithm 3 with $\mathcal{O}_c = \mathcal{O}$ all the time. We show that any node in V_f must be in the region $\mathcal{R}_f$ constructed in Section 3.5.

Lemma 3.6.7. *If $x \in V_f$, then $x \in \mathcal{R}_f$.*

Proof. Since the grid size of GL is l, we have $G_f \subset GL$, meaning all vertices in G_f are grid points on GL. In fact every node generated by the algorithm must be a grid point on GL. By the definition of $\mathcal{R}_f$, we only need to prove that $V_o \subset \mathcal{R}$, where V_o is the set of vertices which are chosen to generate new nodes by Algorithm 3. For convenience, we call a node in V_o an interior point of G_f. We call x the ancestor of y if $y \in Nb_V(x)$, the neighborhood of x, and x is generated earlier than y by Algorithm 3. Correspondingly we call y the child of x. We call Algorithm 3 in gradient steps if the potential of the newly chosen node is lower than its ancestor, otherwise we call it in diffusion steps. We use induction to prove the lemma. Because G_f contains a finite number of nodes, the induction process stops after finite steps. At the first step, $x_s \in V_o$ and $x_s \in \mathcal{R}$. We assume that the first k generated nodes in G_f are contained in $\mathcal{R}$. We want to prove that the node $z \in \mathcal{R}$, where z is the node to generate new points at $(k+1)$-th step. We assume w is the ancestor of z that generates w. Obviously, $w \in \mathcal{R}$ by our induction assumption.

1. Gradient case ($p(z) < p(w)$): by the construction of R in Section 3.5.1, to prove $z \in \mathcal{R}$, we only need to show $\rho_t(z, T) > 0$ for some T. There are two scenarios. (1) There exists T_1 such that $\rho(z, T_1) > 0$. In this case, we must be able to find $T_2 \leq T_1$ with $\rho_t(z, T_2) > 0$, because $\rho(z, 0) = 0$ initially. (2) $\rho(z, t) = 0$ for all $t \in [0, T_1]$. Because the algorithm picks the point that has the lowest p value, we have $p(w) - p(z) > p(w) - p(u)$ for all $u \in Nb(w)$, which gives $d_{wz} = 1$ in (3.13). Notice that $w \in \mathcal{R}$, this means there exists T_3 such that $\rho_t(w, T_3) > 0$, which implies that there is a time interval $I = [T_3 - \Delta t, T_3 + \Delta t]$, $\rho_t(w, t) > 0$ for all $t \in I$. Thus, there must exist a T_4 such that $\rho(w, T_4) > 0$. By (3.8), we have $\rho_t(z, T_4) > 0$ if $\rho(z, T_4) = 0$, which implies $z \in \mathcal{R}$.

2. Diffusion case ($p(z) \geq p(w)$): We recall $\mathcal{R}_2 = \bigcup_{s=1}^{W} \mathcal{R}_2^s$. It is easy to see that if $s > t$, we have $p(x) > p(y)$ for arbitrary $x \in \mathcal{R}_2^s$ and $y \in \mathcal{R}_2^t$ because of the property

of the Gibbs' distribution. Otherwise the diffusion procedure for the set construction is stopped according to the definition of W. We split the proof into three steps for this case. Firstly, we let

$$x = \arg\min_{z}\{p(z) : z \in \bigcup_{y\in\mathcal{R}_2^W} Nb(y)\backslash\mathcal{R}_2\}.$$

and $x_p \in Nb(x) \cap \mathcal{R}_2$. By the definition of W, we know that $x \in \mathcal{R}$, and x is the choice for the starting configuration for the next step. And also, we claim that x exists. Since $x_f \in GL$ has the lowest potential and GL is a connected graph, there must be a node in $\mathcal{R}$ such that one of the outside neighbor has a lower potential, otherwise the construction of $\mathcal{R}$ do not stop.

Secondly, we claim that the part of the V_o generated by the diffusion steps of graph generation is contained in $\mathcal{R}_2$, and we simply use V_o to denote the diffusion part generated by the algorithm. Otherwise, there exists $z \in V_o\backslash\mathcal{R}_2$ with its ancestor $w \in \mathcal{R}_2^s$ for some $s < W$. Without loss of generality, we assume z is the first node outside of $\mathcal{R}_2$ chosen by Algorithm 3. Because $p(z) > p(w)$, we have $z \notin \cup_{i=1}^{s}\mathcal{R}_2^i$. Also, z is a candidate for all $\mathcal{R}_2^i$ with $i = s+1, \cdots, W$. Therefore, $z \notin \mathcal{R}_2$ means that $p(z) > p(u)$ for all $u \in \mathcal{R}_2$. By Algorithm 3, this can happen only when $\mathcal{R}_2^o \subset V_o$ where

$$\bar{\mathcal{R}}_2^o = \{u : u \in \mathcal{R}_2, p(u) \leq p(x_p)\},$$

and $\mathcal{R}_2^o \subset \bar{\mathcal{R}}_2^o$ is the connected component of $\bar{\mathcal{R}}_2^o$ containing x_p. Otherwise there exists $v \in Nb_V(a)\backslash V_o$ for some $a \in V_o$ such that $v \in \mathcal{R}_2$. In this situation, the algorithm will choose v instead of z because of $p(v) < p(z)$. This cause a contradiction with the fact z is the next node selected by the algorithm. On the other hand, if $\mathcal{R}_2^o \subset V_o$ holds, we have $x \in V_o$ and there exists $y \in V_o$ such that $p(y) \leq p(x) < p(z)$, leading to the fact that Algorithm 3 does not put z into V_o. Thus, $V_o \subset \mathcal{R}_2 \subset \mathcal{R}$.

At last, we claim that $x \in V_o$, which gives us that the start node of the next gradient step shared by both the construction of $\mathcal{R}$ and the graph generation in the algorithm. That is, x is chosen by Algorithm 3 to generate new points. Otherwise by the same argument for the existence of x, there is some node $y \in V$ such that Algorithm 3 generates a node q based on y with $p(q) < p(y)$ and $p(y) - p(q)$ maximizes the potential gap amongst all similar pairs. So, $y \in V_o$. However, by the definition of x, it is the lowest potential point has this property within the region $\mathcal{R}_2$. Thus $y = x$. And in the next generation step, q will be generated, and meanwhile, the region growing procedure will also choose q as the start point of the next gradient step.

□

Proof. (Proof of Theorem 3.5.1) Let $G_f = (V_f, E_f)$ be the graph generated by the algorithm with $\mathcal{O}_c = \mathcal{O}$ all the time. We denote $G = (V, E)$ a graph generated with currently known environment $\mathcal{O}_c$ and initial configuration at x. Since $x \in G_f$ at the first step, a simple induction argument can show that every initial configuration x used to generate the graph G must satisfy $x \in G_f$ too. We claim that G and G_f must be the same in the ball $B(x, R)$, where we recall R the detectable radius, i.e. $G \cap B(x, R) = G_f \cap B(x, R)$, because G and G_f are generated by the same algorithm with the same knowledge of the environment in $B(x, R)$. We denote the connected common part of G and G_f as G_o, and further claim that the path γ is in G_o. If not, there must exist the first node generated in the graph generation step $y \in G \backslash G_o$ on the feasible trajectory on G. This implies that $y \in \mathcal{O} \backslash \mathcal{O}_c$ since the ancestor of y is in G_o, which can not contain any node in the infeasible region. However when the robots move along the path γ to the node before y, the system can detect that y is infeasible because $R > L$, and should stop before reaching y, which contradict with the fact that y is on the feasible trajectory. This concludes that the trajectory of the robots is restricted on $G_o \subset G_f$ in arbitrarily given known environment $\mathcal{O}_c \subset \mathcal{O}$. By Lemma 3.6.7, G_f is bounded by $\mathcal{R}_f$, hence the trajectory of the system is bounded by $\mathcal{R}_f$. □

3.7 Conclusion

In this chapter, an iterative algorithm is presented to solve the pathfinding problem in unknown environments. The algorithm contains Graph Generating, Path Finding and Environment Updating steps, among which graph generating is the key one. Guided by a potential function, the algorithm originates from an initial configuration, and expands a tree graph, aiming to connect to the target configuration. Our approach has several advantages. First of all, the algorithm is deterministic and has proven convergence in finite steps. Secondly, the graph generating algorithm has a linear growth rate with respect to the dimension of the configuration space. Together with the dimension reduction strategies for escaping the local traps, our algorithm is suitable to solve high dimensional problems. Lastly, due to the relation to FPE and optimal transport theory, only part of the configuration space needs to be searched. We also remark that our assumption on the existence of a feasible path is only a technical requirement for the proof of the convergence. If such a path does not exist, the proposed algorithm can recognize the situation and stop the calculation in a finite number of iterations.

The proposed algorithm is our initial exploration of using optimal transport theory for path planning. Further improvements can be made on many fronts. For example, one can speed up the calculation in escaping the traps at local minimizers, by using random graph generating strategies. This is very useful in high dimensional situations. The graph generating radius l can be adaptive. Longer steps can be taken in wide open space and shorter step size is used when encountered narrow corridors. In addition, the proposed algorithms can be adapted to construct a general strategy for control problems with constraints, especially when some of the constraints are only available during the process.

CHAPTER 4
DYNAMICAL PATH PLANNING METHOD FOR CONTROL PROBLEMS WITH PARTIALLY GIVEN CONSTRAINTS

4.1 Introduction

In the chapter, we generalize the algorithm and theory to the control theory setup, especially when some of the constraints are not known a prior. In this case, feasible path acquisition is the only demand while one does not care about the optimality, many path planning algorithms can be used such as those mentioned in Section 3.1. Given the time-invariant continuous control system $(\Omega, \mathcal{U}, f)$ where Ω is the state space, $\mathcal{U}$ the control space and f the dynamical system with holonomic constraints ϕ, we have

$$\dot{x} = f(x, u)$$
$$\phi(x(t)) > 0$$
$$x(t) \in \Omega, u(t) \in \mathcal{U} \quad \forall t$$

If for some state x, initially we have $\phi(x) \geq 0$, which means x is a feasible state, however when certain condition holds, as an example, given a trajectory γ, at time t, the distance $\gamma(\tau)$ and x is small enough for some $\tau \leq t$, x then is no longer feasible ($\phi(x) < 0$). In the setup where all the constraints information is provided, we can define specific objective function to calculate the optimal path using various of methods. However when we do not have all constraints a prior, optimality is not defined any more. In this chapter, we study the problem described above and generalize the algorithm in Chapter 3 to search for a feasible path combined with a corresponding set of control variables. The algorithm has the same

three main steps: Graph Generation, Path Finding and Trajectory Extension. Among them, Graph Generation is the core part, which is guided by the evolution of the Fokker-Planck equation (FPE) in the Optimal Transport Theory. This chapter is organized as following: In Section 4.2, we formulate the problem formally and provide our algorithm. Within this section, we discuss the major steps of our algorithm in detail. Specially in Section 4.2.1, we give a general algorithm for graph generation and provide specific graph growing strategies if the control system can form a Riemannian manifold or even is flat locally. Later in Section 4.3, we carry out several experiments to show how the algorithm works and Section 4.4 shows the relation between our algorithm and FPE. At last, in Section 4.5, we give the concrete proof for the theorems.

4.2 The Algorithm

We consider a complete control system $(\Omega, \mathcal{U}, f)$ in the state space Ω, a compact d dimensional Riemannian manifold equipped with a Riemannian metric $g(\cdot, \cdot)$, which induces a distance function $d(\cdot, \cdot)$. Also, we take the control space $\mathcal{U}$ to be a compact subset of $\mathbb{R}^n$ and let the dynamics $f(x, u)$ be a Lipschitz function, that is,

$$g(f(x, u), f(y, v)) \leq C\,(d(x, y) + \|u - v\|)$$

for arbitrary $x, y \in \Omega$ and $u, v \in \mathcal{U}$. Accompanied with the system, there are two types of continuous constraints to restrict the state space denoted as ϕ and ψ, which are the same as Section 3.2 where ϕ is the given constraint while ψ is the constraint to be discovered while in motion. We introduce the same notation $\hat{\psi}$ to be the actual known constraint for ψ up to time t and the definition is provided as (3.1).

Our goal is to find a control function $u \in \mathcal{U}^{[0,T)}$ such that the induced path $\gamma \subset \Omega$

satisfies

$$\begin{aligned}
&\dot{\gamma}(t) = f(\gamma(t), u(t)),\\
&\gamma(0) = x_0, \gamma(T) = x_f,\\
&\phi(\gamma(t)) \geq 0 \text{ for all } t \in [0, T],\\
&\hat{\psi}(\gamma(t), \gamma, t) \geq 0 \text{ for all } t \in [0, T],
\end{aligned}$$

where x_0 is the given initial state, x_f is the target state and $\psi(x) \geq 0$ ($\phi(x) \geq 0$) means that for all $i = 1, \cdots, k_2(k_1)$, $\psi_i(x) \geq 0$ ($\phi_i(x) \geq 0$). Further, we assume that x_f is controllable from any $x \in \Omega$ in time T and there exists $\omega \in \mathcal{U}$ such that $f(x_f, \omega) = 0$.

On the other hand, if a state x violate the constraints, say $\psi(x) < 0$ ($\phi(x) < 0$), then there exists $k \in \{1, \cdots, k_2(k_1)\}$ such that $\psi_k(x) < 0$ ($\phi_k(x) < 0$). Using this, we define the infeasible set $\mathcal{O}$ and the currently known infeasible set with respect to the path γ at time t $\mathcal{O}_c(\gamma, t)$ (if γ and t are clear, we will omit them and directly denote the currently known infeasible region as $\mathcal{O}_c$ later in this chapter) in the space as

$$\begin{aligned}
\mathcal{O} &= \{x \in \Omega : \phi(x) < 0 \text{ or } \psi(x) < 0\},\\
\mathcal{O}_c(\gamma, t) &= \left\{x \in \Omega : \phi(x) < 0 \text{ or } \hat{\psi}(x, \gamma, t) < 0\right\},
\end{aligned}$$

where $\mathcal{O}$ are open since ϕ and ψ are continuous. In addition, we denote the boundary of the infeasible set to be $\partial\mathcal{O}$ and $\partial\mathcal{O}_c(\gamma, t)$ and introduce the feasible path from x_0 to x_f set as

$$\begin{aligned}
\Gamma = \Big\{&\gamma \subset \Omega : \gamma(0) = x_0, \gamma(t) = x_f, \forall t \in [T_0, T] \text{ for some } T_0 \leq T,\\
&\dot{\gamma} = f(\gamma, u) \text{ where } u \in \mathcal{U}^{[0,T)}, \gamma \cap \mathcal{O} = \emptyset\Big\}
\end{aligned}$$

At last, we assume that

$$\sup_{\gamma \in \Gamma} \inf_{t \in [0,T]} \sup_{r \geq 0} \{r : B(\gamma(t), r) \cap \mathcal{O} = \emptyset\} = L > 0, \tag{4.1}$$

where $B(x,r)$ is the ball and corresponding sphere centered at x with radius r as

$$B(x,r) = \left\{ \begin{array}{l} y : \int_0^t \sqrt{g(\dot{\gamma},\dot{\gamma})} d\tau < r \text{ where } \gamma(0) = x, \\ \quad \gamma(t) = y, \dot{\gamma} = f(x,u) \text{ for some } u \in \mathcal{U}^{[0,t)} \end{array} \right\},$$

$$S(x,r) = \partial B(x,r).$$

This defines a tubular feasible region $\mathcal{T}$ with radius L. With the assumption that $x \in B(x_f, L)$ can be controlled to x_f without leaving $B(x_f, L)$, we give Algorithm 4 to handle the problem.

Algorithm 4: Feasible Path Finding

Data: initial state x_s, target state x_t, initial known constraints $\mathcal{O}$
Current state $x = x_0$
Current known constraints $\mathcal{O}_c = \mathcal{O}$
$\gamma_0(0) = x_0, T = 0$
while $x \neq x_f$ **do**
    Graph Generation: Generate a connected graph G containing x_s, x_t with all edges and vertices not in $\mathcal{O}_c$
    Path Finding: Find a (shortest) path γ on G from x_s to x_t and the corresponding control $u \in \mathcal{U}^{[0,T_1)}$
    Trajectory Extension: Let $\zeta = \gamma_0$ and $\zeta(T+t) = \gamma(t)$ for $t \in [0, T_1]$
    if $\phi(\gamma(t)) \geq 0$ *and* $\hat{\psi}(\gamma(t), \zeta, T+t) \geq 0$ *for all* $t \in [0, T_1]$ **then**
        $\gamma_0 = \zeta$
        $T = T + T_1$
        $x = x_f$
    else
        Find T_2 such that $\phi(\gamma(t)) \geq 0$ and $\hat{\psi}(\gamma(t), \zeta, T+t) \geq 0$ for all $t \in [0, T_2]$
        $\gamma_0 = \zeta([0, T+T_2])$
        $T = T + T_2$
        $x = \gamma(T_2)$
    end
    return γ_0
end

In the rest of this section, we will concentrate on the three major steps in the while loop of Algorithm 4 and give detailed explanation with examples on how to carry out those procedures.

4.2.1 Graph Generation

The first and most important step of the algorithm is generating a graph $G = (V, E)$ with x_0, x_f in the same connected component, where V denotes the vertex set containing node lying in the currently feasible region. During the graph generation step, since the current feasible trajectory γ_0 does not change, the constraints keeps the same and we denote the currently known infeasible region to be $\mathcal{O}_c$, which is initially set to be

$$\mathcal{O}_c = \{x : \phi(x) < 0\}$$

or is updated in the manner mentioned in Section 4.2.3, we have $V \subset \Omega \backslash \mathcal{O}_c$. To form the edges on the Riemannian manifold Ω, we introduce the map $\Phi(x, u, r) = \gamma_{u,x}(T_0)$ where

$$\begin{aligned} &\dot{\gamma}_{u,x}(t) = f(\gamma_{u,x}(t), u(t)) \\ &\int_0^{T_0} \sqrt{g(\dot{\gamma}_{u,x}, \dot{\gamma}_{u,x})} dt = r \\ &\gamma_{u,x}(0) = x \end{aligned}$$

Denoting $\gamma_{u,x} = \gamma_{u,x}([0, T_0])$ the segment linking x and $\Phi(x, u, r)$, we give the following procedure to generate the graph:

We initially set $V = \{x_0\}$, the edge set $E = \emptyset$ and introduce a convex potential function with target being the unique minimizer in Ω. Here one choice of the potential is $p(x) = d(x, x_f)$. In each step, we perform the following procedure until get a graph connecting x_0 and x_f:

- We choose a vertex $y \in V$ with the lowest potential, meaning that $y = \arg\min_{x \in V} p(x)$.
- For the chosen node y, we select a set of control functions $N = \{u_i\}_{i=1}^c$ with $u \in \mathcal{U}^{[0,T)} \subset \mathcal{U}^{[0,T)}$. With this, we can have the vertex candidates as $\{\Phi(y, u_i, l_i(y))\}_{i=1}^c$ and the edge candidates to be $\{\gamma_{u_i,y}\}_{i=1}^c$.

- If $\Phi(y, u_i, l_i(y)) \in \mathcal{O}_c$ or $\gamma_{u_i,y} \cap \mathcal{O}_c \neq \emptyset$, we eliminate $\Phi(y, u_i, l_i(y))$ from the candidate set and corresponding edges from candidate edge set.

- We delete $\Phi(y, u_i, l_i(y))$ if there is $z \in V$ such that $d(\Phi(y, u_i, l_i(y)), z) \leq \epsilon$ for some prior set parameter ϵ.

- If some of the newly added nodes are close enough to the target x_f, that is, $d(z, x_f) \leq L$ where z is newly added, from z to x_f we compute a path that never leaves $B(x_f, L)$, add x_f in V and terminate the graph generating procedure.

As an example, Figure 4.1 gives the feasible and infeasible regions on the upper semi sphere Ω where the obstacles are in dark gray with initial and target states marked in red. On the sphere, we set the dynamics to be $\dot{x} = u$, which forms Ω a compact manifold and we will discuss more about this case in later. Every time, we pick the lowest-potential node and generate geodesics as our edges (depicted in Figure 4.1(a)). However, there exists constraints to consider. Thus, we need to delete corresponding candidates as described above and is shown in Figure 4.1(e), 4.1(f), 4.1(c), 4.1(d), 4.1(g) and 4.1(h). At last, Figure 4.1(b) displays the final graph on the semi sphere case.

To ensure the convergence of the algorithm, we further give restriction on the generated candidate nodes at the node x that

$$B(x, L) \subset \bigcup_{y \in \mathcal{F}(x)} B(y, L), \tag{4.2}$$

where

$$\mathcal{F}(x) = \{\Phi(x, u_i, l_i(y)) : u_i \in N, d(\Phi(x, u_i, l_i(y)), z) \geq \epsilon \text{ for all } z \in V\}. \tag{4.3}$$

This condition is used for the proof of convergence and is the guideline to choose controls, but does not need to be followed strictly while performing the algorithm. Now we summarize all the strategies mentioned and give the graph generating procedure with the currently

known infeasible set $\mathcal{O}_c$ as Algorithm 5.

Algorithm 5: Graph Generation

Data: The initial state x_0, target state x_f, the potential function p, currently known infeasible region $\mathcal{O}_c$

Result: G=(V,E,p)

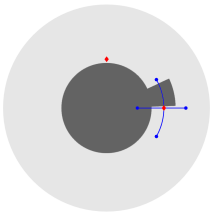
$V = \{x_0\}, Q = V, E = \emptyset$
while $t \notin V$ **do**
  $point\_add = False$
  while *not* $point\_add$ **do**
    $x = \arg\min_{z \in V} p(z)$
    if $p(x) < +\infty$ **then**
      $K = \{\Phi(x, u, l(u)) : u \in N, \gamma_{u,x} \cap \mathcal{O}_c = \emptyset, d(\Phi(x, u, l(u)), z) \geq \epsilon(\forall z \in V)\}$ with N satisfies (4.2)
      $K = K \backslash V$
      $V = V \cup K$
      $E = E \cup \{\gamma_{u,x} : u \in N, \Phi(x, u, l(u)) \in K\}$
      for $q \in K$ **do**
        if $d(q, x_f) \leq L$ **then**
          $V = V \cup \{x_f\}$
          $E = E \cup \{\gamma\}$ where $\gamma \subset B(x_f, L)$ is the segment linking q, x_f with $\dot{\gamma} = f(\gamma, u)$ for some $u \in \mathcal{U}^{[0,T)}$
        end
      end
    else
      return $G = \emptyset$
    end
  end
end
return $G = (V, E, p)$

Reversible control, a special case

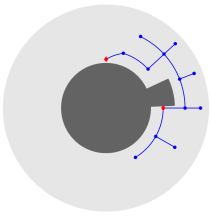
When the control system is strongly reversible [1], we can further give an explicit strategy to generate points originated at node x. We define

$$\hat{d}(x, y) = \inf_{\gamma \subset \Omega} \left\{ \int_0^{T_0} \sqrt{g(\dot{\gamma}, \dot{\gamma})} dt : \dot{\gamma} = f(\gamma, u) \exists u \in \mathcal{U}^{[0,T_0)}, \gamma(0) = x, \gamma(T_0) = y \right\},$$

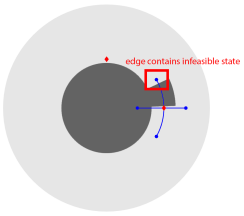
[1] A control system $(\Omega, \mathcal{U}, f)$ is strongly reversible if for all $x \in \Omega$ and each $u \in L^\infty_{\mathcal{U}}(0, T)$ for x, there exists $v \in L^\infty_{\mathcal{U}}(0, T)$, such that when denoting the path generated by u from x is $\gamma(t)$ with $t \in [0, T]$, the trajectory generated by v from $\gamma(T)$ is $\zeta(t) = \gamma(T - t)$.



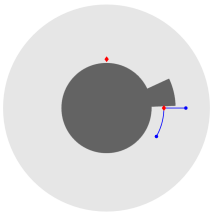
(a) The red diamonds are initial and target states while the gray portion are the infeasible region. The algorithm generate 4 new nodes around the initial state in the first step.



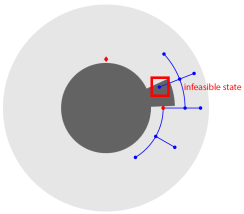
(b) The final graph generated in the given feasible region.



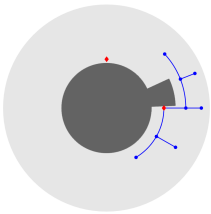
(c) The algorithm generates a new node candidate whose edge linking to its ancestor is across the infeasible region.



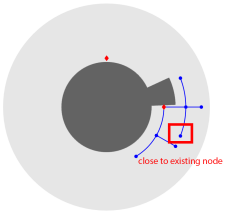
(d) The graph after deleting the node with edge blocked in Figure 4.1(c).



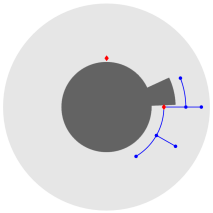
(e) The algorithm generates a new node candidate that is in the infeasible region and should be deleted.



(f) The graph after eliminating the infeasible node in Figure 4.1(e).



(g) The algorithm generates a new node candidate that is too close to an existing vertex.



(h) The graph after deleting the too close node in Figure 4.1(g).

Figure 4.1: This plot shows the graph generation step. Every time the algorithm picks a lowest-potential node and generates candidates around it. After deleting infeasible nodes, vertices with infeasible edges and nodes that are too close to existing vertices, the left candidates are contained in the vertex set with their edges linking to their ancestor in the edge set.

where $T_0 = T_0(\gamma)$ is the first time such that $\gamma(T_0) = y$.

Proposition 4. *$\hat{d}$ is a distance function on Ω if $(\Omega, \mathcal{U}, f)$ is strongly reversible.*

Proof. It is obvious that $\hat{d}(x, x) = 0$ and triangular inequality holds. For symmetry property, we first let

$$\mathcal{L}(\gamma) = \int_0^{T_0} \sqrt{g(\dot{\gamma}, \dot{\gamma})} dt$$

and

$$\mathcal{S}(x, y) = \left\{ \gamma \subset \Omega : \dot{\gamma} = f(\gamma, u) \text{ for some } u \in \mathcal{U}^{[0,T_0)}, \gamma(0) = x, \gamma(T_0) = y \right\}.$$

For each $\gamma \in \mathcal{S}(x, y)$, letting T_0 be the time when $\gamma(T_0) = y$, we have a corresponding control $v \in \mathcal{U}^{[0,T_0)}$ and trajectory $\hat{\gamma}$ such that $\hat{\gamma}(t) = \gamma(r(T - t))$ where $r(\cdot)$ is a re-parameterization of time with $r' > 0$. Then $\mathcal{L}(\gamma) = \mathcal{L}(\hat{\gamma})$. Let $\{\gamma_i\}_{i=1}^k \subset \mathcal{S}(x, y)$ be the set that $\hat{d}(x, y) = \lim_{i\to\infty} \mathcal{L}(\gamma_i) = \lim_{i\to\infty} \mathcal{L}(\hat{\gamma}_i)$. If $\hat{d}(y, x) < \lim_{i\to\infty} \mathcal{L}(\hat{\gamma}_i)$, then there exits $\hat{\zeta} \in \mathcal{S}(y, x)$ such that $\lim_{i\to\infty} \mathcal{L}(\hat{\gamma}_i) > \mathcal{L}(\hat{\zeta})$. But since the corresponding $\zeta \in \mathcal{S}(x, y)$, we can have that

$$\hat{d}(x, y) = \lim_{i\to\infty} \mathcal{L}(\gamma_i) > \mathcal{L}(\zeta).$$

which gives the contradiction. Therefore,

$$\hat{d}(y, x) = \lim_{i\to\infty} \mathcal{L}(\hat{\gamma}_i) = \lim_{i\to\infty} \mathcal{L}(\gamma_i) = \hat{d}(x, y).$$

Thus, $\hat{d}$ is a distance function. □

With $\hat{d}$, we can get an induced Riemannian metric $\hat{g}$ and select a set of orthonormal basis $\{e_i\}_{i=1}^d$, that is, $\hat{g}_x(e_i, e_j) = \delta_{ij}$ at any state x, and let $N = \{\pm e_i\}_{i=1}^d$. We introduce the exponential map $\exp(x, w, t)$ as the geodesic starting at the state x with w be the initial velocity, i.e. $\frac{d\exp}{dt}(x, w, 0) = w$ where the third parameter t is the time parameter. Next, we re-parameterize the exponential map to introduce the arc length parameter and rewrite

the map to be $\exp(x, w, s)$. Lastly, we form the vertex candidates set as $\{\exp(x, w, l)\}$ and edge candidates set as $\{\exp(x, w, [0, l])\}$ where $w \in N$ and l is the node generating radius. After getting the candidates sets, we perform the same deleting strategy in Section 4.2.1: we delete the candidate nodes if they or their linking edges activate the constraints; Further, we delete those nodes that are on the position there already is a node generated before. To get the control for each edge, one needs to solve $\dot{\gamma} = f(\gamma, u)$, where $\gamma(t) = \exp(x, w, t)$, at each time step, that is, $u = (f(x, \cdot))^{-1}(\dot{\gamma})$. Although there can be various of controls for a certain path γ, only one of them is desired for our problem. Also for the convergence concern, we restrict l to satisfy

$$B(x, L) \subset \bigcup_{w \in \mathcal{F}(x)} \exp_x (B(lw, L)) \tag{4.4}$$

where $\mathcal{F}(x)$ is defined in (4.3). Furthermore, if Ω is a Lie group with the Lie bracket vanishes (i.e. $[X, Y] = 0$ for arbitrary vector field X, Y), then we have

$$l \in (0, \frac{2L}{\sqrt{\lambda d}}]$$

where λ is the largest eigenvalue of the metric $\hat{g}$. However, this is a condition for the theoretical convergence. In practice, one can adjust the generating radius according to the actual feasible region. In this special case, the algorithm is as Algorithm 6.

4.2.2 Path Finding

To find a feasible (optimal) path on the generated graph $G = (V, E)$, we use the back-tracing strategy discussed in Section 3.2.2. Since the same procedure and properties can be directly used here, we skip the details in this part.

Algorithm 6: Graph Generation For Strong Reversible System

Data: The initial state x_0, target state x_f, the potential function p, currently known infeasible region $\mathcal{O}_c$, generating radius l and an orthonormal generating direction set $N = \{\pm e_i\}_{i=1}^{d}$

Result: G=(V,E,p)

$V = \{x_0\}, Q = V, E = \emptyset$
while $t \notin V$ **do**
  $point\_add = False$
  while *not* $point\_add$ **do**
    $x = \arg\min_{z \in V} p(z)$
    if $p(x) < +\infty$ **then**
      $K = \{\exp(x, w, l) : w \in N, \exp(x, w, [0, l]) \cap \mathcal{O}_c = \emptyset\}$
      $K = K \backslash V$
      $V = V \cup K$
      $E = E \cup \{\exp(x, w, [0, l]) : w \in N, \exp(x, w, l) \in K\}$
      for $q \in K$ **do**
        if $d(q, x_f) \leq L$ **then**
          $V = V \cup \{x_f\}$
          $E = E \cup \{\exp(q, w, s)\}$ for some $w \in T_q\Omega$ and some positive number s with $\exp(q, w, [0, s]) \subset B(x_f, L)$
        end
      end
    else
      return $G = \emptyset$
    end
  end
end
return $G = (V, E, p)$

4.2.3 Trajectory Extension

Once the feasible trajectory $\gamma([0, T_1])$ is found on the generated graph, a new trial along the path to reach the target can be conducted. We let the previous feasible path segment to be $\gamma_0([0, T_0])$ as denoted in Algorithm 4. First of all, we extend γ_0 with the new path γ by introducing ζ where

$$\zeta(t) = \begin{cases} \gamma_0(t) & t \in [0, T_0] \\ \gamma(t - T_0) & t \in [T_0, T_0 + T_1] \end{cases},$$

and check whether for all $t \in [0, T_1]$, $\phi(\gamma(t)) \geq 0$ and $\hat{\psi}(\gamma(t), \zeta, t + T_0) \geq 0$. If so, we let $\gamma_0 = \zeta$ and a feasible path from x_0 to x_f is found. Otherwise, there exists a time t such that $\phi(\gamma(t)) < 0$ or $\hat{\psi}(\gamma(t), \zeta, T_0 + t) < 0$ and we can set

$$T_2 = \inf_t \left\{ t : \phi(\gamma(t)) < 0 \text{ or } \hat{\psi}(\gamma(t), \zeta, T_0 + t) < 0 \right\}.$$

We introduce the stopping time $T_s < T_2$ such that $d(\gamma(T_s), \gamma(T_2)) < R$, where R is defined in (3.1). In the algorithm, we can actually have T_s and stop at $\gamma(T_s)$ before arriving the position $\gamma(T_2)$ because of the following property:

Proposition 5. *Let $B_d(x, r) = \{y : d(x, y) < r\}$ and R defined in* (3.1)*. There exists $t > T_2$ with $\gamma(t) \in B_d(\gamma(T_s), R)$ that $\phi(\gamma(t)) < 0$ or $\hat{\psi}(\gamma(t), \zeta, T + T_s) < 0$.*

Proof. Since $d(\gamma(T_s), \gamma(T_2)) < R$, by the continuity of γ, which is from the construction strategy of γ in our algorithm, there exists $I = (s_1, s_2)$ such that $T_2 \in I$ and $\gamma(I) \subset B_d(\gamma(T_s), R)$. By the definition of T_2, we can find $t \in (T_2, s_2)$ such that $\phi(\gamma(t)) < 0$ or $\hat{\psi}(\gamma(t), \zeta, T_0 + t) < 0$. If $\phi(\gamma(t)) < 0$, then the proof ends.

On the other hand if $\phi(\gamma(t)) > 0$, $\hat{\psi}(\gamma(t), \zeta, T_0 + t) < 0$ holds. Notice that $\gamma(t) \in B_d(\gamma(t), R)$, we have $\psi(\gamma(t)) = \hat{\psi}(\gamma(t), \zeta, T_0 + t) < 0$. Meanwhile, since $t \in I$, $\gamma(t) \in B_d(\gamma(T_s), R)$, which leads to $\hat{\psi}(\gamma(t), \zeta, T_0 + T_s) = \psi(\gamma(t)) < 0$, with which, we can end

our proof. □

After stopping at $\gamma(T_s)$, we further update $\gamma_0 = \zeta([0, T_0 + T_s])$, $T_0 = T_0 + T_s$, then the currently infeasible set to be

$$\mathcal{O}_c = \left\{ x : \phi(x) < 0 \text{ or } \hat{\psi}(x, \gamma_0, t) < 0 \text{ for } t \in [0, T_0] \right\}.$$

At last, we go back to the graph generation step. Each time a new path γ_i is produced when the current path is blocked. We collect all paths produced in Algorithm 5 as $\{\gamma_i\}_{i=1}^m$, and their stopping time set as $\{T_i\}_{i=1}^m$. Same as Chapter 3, from the choices of stopping time, we may require that there exists a non-negative constant q satisfying $q < R$, and for all $\epsilon > q$, $B_d(\gamma_i(T_i), \epsilon)$ has non-empty intersection with $\mathcal{O}_c^i$ for every $i = 1, \cdots, m$, where $\mathcal{O}_c^i$ denotes the updated $\mathcal{O}_c$ at step i. Such selected stopping time set satisfies the following property

$$\sup_i \inf_\epsilon \left\{ \epsilon : B_d(\gamma_i(T_i), \epsilon) \cap \mathcal{O}_c^i \neq \emptyset \right\} = q < R, \tag{4.5}$$

4.2.4 Convergence and Complexity

The proposed algorithm (Algorithm 4) converges in finite steps. To see this, since path finding on a graph $G = (V, E)$ takes $O(|E| + |V|) < \infty$ when $|V|, |E| < \infty$, one needs to show that the graph generation takes finite number of steps and the number of trajectory extension is finite, which are implied by the following theorems.

Theorem 4.2.1. *If* (4.1) *and* (4.2) *hold, then Algorithm 5 stops in finite steps. That is, the loop in the algorithm terminates in a finite number of steps, the generated graph* $G = (V, E, K, p)$*, which entirely lies the current feasible set* $\Omega \backslash \mathcal{O}_c$ *is connected and has a finite number of vertices* $|V| < \infty$*. Furthermore,* $x_0, x_f \in V$ *if* $\Gamma \neq \emptyset$*.*

Theorem 4.2.2. *If* (4.1) *holds and* $(\Omega, \hat{g})$ *is compact, then Algorithm 6 converges in finite steps in the same sense as Theorem 4.2.1.*

Theorem 4.2.3. *Let $\{\gamma_i\}_{i=1}^m$ be the paths produced by Algorithm 4 with $\{T_i\}_{i=1}^m$ being the stopping time set. If the assumptions in Theorem 4.2.1 (assumptions in Theorem 4.2.2 holds if Algorithm 6 is used for graph generation) and* (4.5) *hold, then $m < \infty$.*

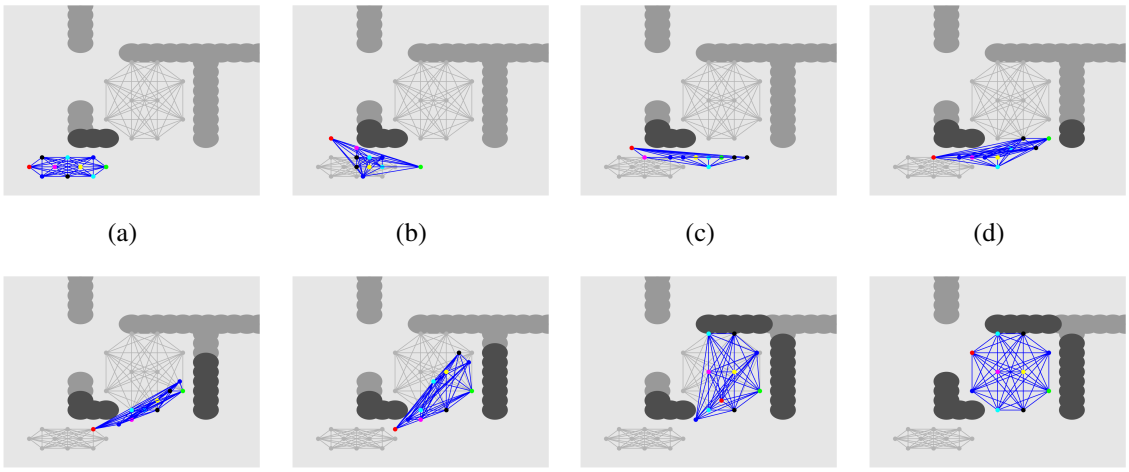
4.3 Experiments

Section 3.3 exhibits examples when the manifold is flat. In this section, we conduct several experiments with different setups. We will show that our algorithm can handle high dimensional cases with an example of multi-robots moving in an environment with locations of the obstacles stay unknown unless one of the robots is close to it. At last, on the upper semi sphere, we give the results of our algorithm to show how it works on the manifold.

4.3.1 High Dimensional Experiment on Euclidean Space

In this experiment, we move 10 robots in the $[0, 1] \times [0, 1]$ square with each two robots must maintain a distance between 0.03 and 0.5. Since the working space is 2-dimensional, the state space Ω is 20 dimension. In this square, there are certain positions that are infeasible for all of the robots, which forms the unknown infeasible set of Ω. Same as is in Section 3.3, light gray and dark gray correspondingly represent the invisible and visible infeasible set. Furthermore, we link each two robots with a straight line in the square and impose the restriction that every point on the straight lines cannot be in the infeasible positions either. Meanwhile, we perform node deleting procedure only when two nodes are of same state. With these known and unknown constraints, the system originated at the bottom left corner in a diamond shape, aiming to reach the center of the square. Figure 4.2 gives parts of the shapes while the system is moving. We can see that the system twisted a little bit and after recognizes the infeasible part in front of it, the robot system keeps itself away from that part and then gradually forms the desired shape in the target state. The video of the motion of the system can be found online [2].

[2]The video can be found in https://youtu.be/WTl9tgINgqk



(a) (b) (c) (d)

(e) (f) (g) (h)

Figure 4.2:

4.3.2 Example on Upper Semi Sphere

At last, we conduct the experiment on the upper semi sphere with the metric tensor being

$$ds^2 = d\phi^2 + \sin^2(\phi)d\theta^2,$$

which is in the spherical coordinates format and

$$\begin{cases} x = \sin\theta\cos\phi \\ y = \sin\theta\sin\phi \\ z = \cos\theta \end{cases},$$

with x, y, z being the Cartesian coordinates. Since Ω is the upper part of the sphere, $\theta \in [0, \pi/2]$ and $\phi \in [0, 2\pi)$. With this metric, we can easily calculate the geodesic distance between two points $x = (\theta_1, \phi_1)$ and $y = (\theta_2, \phi_2)$:

$$d(x, y) = 2\pi \arccos(\sin\theta_1\cos\phi_1\sin\theta_2\cos\phi_2 + \sin\theta_1\sin\phi_1\sin\theta_2\sin\phi_2 + \cos\theta_1\cos\theta_2).$$

To select the nodes generation strategy, we use geodesics with arc length l as our edges and the endpoints to be the vertex candidates. Although in general, the geodesics calculation involves solving ordinary differential equations, we choose the coordinate directions under spherical coordinates, which gives us the candidates nodes around a certain point $(\theta, \phi) \in \Omega$ as $(\theta \pm \Delta\theta, \phi)$ and $(\theta, \phi \pm \Delta\phi)$. And since the geodesic segments are of length l, we can solve for the $\Delta\theta$ and $\Delta\phi$ as

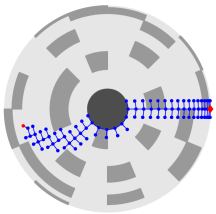
$$\Delta\theta = \frac{l}{2\pi}, \Delta\phi = \frac{l}{2\pi \sin\theta}.$$

Now, we set $l = 0.5$ and define two nodes $x, y \in V$, where V is the vertex set of the generated graph $G = (V, E)$, are too close if $d(x, y) < 0.2$. Depicting the infeasible set in gray blocks in Figure 4.3, the central disk centered at the north pole is known in advance while other parts of infeasible set are unknown. Figure 4.3(a) and 4.3(b) gives the graphs generated during the whole process and Figure 4.3(c) and 4.3(d) are the trajectories. As we can see, at first, since only part of the infeasible set is known, the graph mainly moves to the target directly. However, when detected new infeasible states, the algorithm searches around the local minimizer and finally get to the target state. At last, we give the final path in Figure 4.3(e).

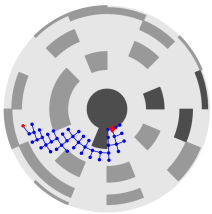
It is worth mentioning that in all three examples, the algorithm only searches part of Ω and tends to expands the graph greedily to the target, which occurs not by accident. As a matter of fact, this algorithm is guided by the Fokker-Planck equation, which can be seen as a gradient flow under 2-Wasserstein metric and we will give detailed discussion about the relation between them in the next section.

4.4 Relationship With FPE and Optimal Transport

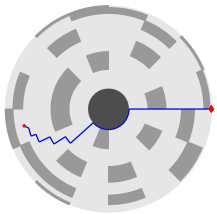
The design of the graph generating algorithm is inspired by the evolution of the density function along the Fokker-Planck equation (3.6). Reversely, we can show that the graph



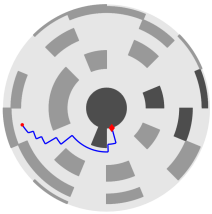
(a) The first graph generated by the algorithm. Since only part of the infeasible set is known, the graph mainly moves to the target directly.



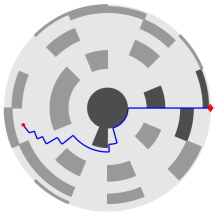
(b) The second graph generated after the algorithm notices the original trajectory is infeasible.



(c) The trajectory searched on the first graph.



(d) The trajectory calculated on the second graph.



(e) The final trajectory

Figure 4.3: This set of plots shows the graphs and trajectories generated on the upper semi sphere with part of the region infeasible.

generation is guided by FPE. If the infeasible set $\mathcal{O}$ is given, the vertex set of the graph generated by the graph generating algorithms (Algorithm 5 and 6) is bounded by the node set given explicitly by evolving FPE with specific initial conditions. If we set our algorithm to stop only when there is no new node can be added, the graph generating strategies give a method to cover $\Omega\backslash\mathcal{O}$ with a tree graph in the sense of

$$\Omega\backslash\mathcal{O} \subset \bigcup_{x\in V} B(x, L)$$

where L is defined in (4.1). However with (3.6), we can describe a subset $\mathcal{R}$ of V such that $G_0 = (V_0, E_0)$ constructed by Algorithm 5 and 6 (we will call them algorithms in the rest of this section) has the property that $V_0 \subset \bar{\mathcal{R}}$ where

$$\bar{\mathcal{R}} = \{x : x \in \mathcal{R} \text{ or } x \in Nb(y) \text{ for some } y \in \mathcal{R}\}.$$

To depict $\mathcal{R}$, we discretize (3.6) on G as follow [52]:

$$\frac{\partial \rho_j}{\partial t} = \sum_{k \in Nb(j)} (F_k(\rho, \beta) - F_j(\rho, \beta))_+ \rho_k - \sum_{k \in Nb(j)} (F_j(\rho, \beta) - F_k(\rho, \beta))_+ \rho_j, \tag{4.6}$$

where $(\cdot)_+ = \max(\cdot, 0)$, $\rho_j = \rho(x_j, t)$, $Nb(j)$ is the set of all adjacent nodes (neighbors) of node x_j on G and $F_j(\rho, \beta) = \frac{\partial}{\partial \rho_j} \mathcal{F}(\rho, \beta)$, in which $\mathcal{F}(\rho, \beta)$ is the free energy

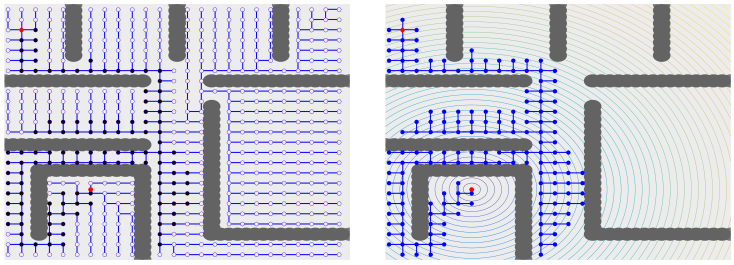
$$\mathcal{F}(\rho, \beta) = \sum_{j=1}^{N} \left(p(x_j)\rho_j + \beta \rho_j \log \rho_j \right).$$

Now with (4.6), we use the same strategy as is in 3.5, changing the evolutionary equation to be (4.6). We iteratively proceed gradient case and diffusion case to expand our region $\mathcal{R}$ and at last refine this region by adding all the neighbors of each node in $\mathcal{R}$ and we have the following theorem:

Theorem 4.4.1. *Suppose $G_0 = (V_0, E_0)$ is the graph generated by Algorithm 5 or 6 with initial and target to be x_s, x_f correspondingly. Providing that $\mathcal{R}$ is constructed by the method described in this section, then $V_0 \subset \bar{\mathcal{R}}$.*[3]

We show the result on $[0, 1] \times [0, 1]$ equipped with Euclidean distance and the trivial dynamics $\dot{x} = u$. The tree $G = (V, E)$ (shown in Figure 4.4(a)) is constructed with $l = 0.04$ which covers the whole feasible space. Followed the evolution of (4.6), the graph $G_0 = (V_0, E_0)$ generated by Algorithm 6 is plotted in Figure 4.4(b). To make the comparison, Figure 4.4(a) offers $\mathcal{R}$, which gives numerical evidence for Theorem 4.4.1 since in Figure 4.4, it is obvious that $V_0 \subset \bar{\mathcal{R}}$. In Figure 4.4(a), the level curve of the potential function is given and we can see that the graph is generated alternatively between the gradient case and the diffusion case, which is guided by the FPE. One thing to notice is that since FPE is evolved on graph, the negative gradient vector is projected onto the directions determined by N.

[3] This can be proven in the same way of the proof for Lemma 3.6.7



(a) The region $\mathcal{R}$ constructed by solving (4.6) on the tree $G = (V, E)$ created by Algorithm two with no termination condition.

(b) The graph generated by Algorithm 6 with the level curves of the potential function given in the background.

Figure 4.4: This figure gives $\mathcal{R}$ and the graph $G_0 = (V_0, E_0)$ generated by Algorithm 6. As we see, $V_0 \subset \bar{\mathcal{R}}$

Remark 3. *In this chapter, FPE is computed on the tree structure directly and gives the guidance of how Algorithm 5 and 6 generate vertices and how far should the graph goes. Different from this perspective, in Chapter 3, the FPE is independent from the graph generation and can also give the bound for the growing of the graph.*

4.5 Convergence Analysis

In this section, we give detailed proof of Theorems stated in Section 4.2.4. Before giving the direct proof, we first address some lemmas. Some of the proofs are the same as in Chapter 3, thus we will only give the Lemmas without providing repeated proof.

Lemma 4.5.1. *If there exists a feasible path, Γ is non-empty and compact with respect to the L^∞ norm given by*

$$d_\Gamma(\gamma_1, \gamma_2) = \sup_{t\in[0,T]} d(\gamma_1(t), \gamma_2(t)),$$

where γ_1 and γ_2 are two paths in Γ, providing that $\mathcal{U}^{[0,T)}$ is also equipped with L^∞ norm

$$d_\mathcal{U}(u_1, u_2) = \sup_{t\in[0,T)} d(u_1(t), u_2(t)),$$

Proof. First of all, it is easy to see that if a feasible path γ_0 exists, it must satisfy $\gamma_0(0) = x_0$

and $\gamma_0(T_0) = x_f$ for some $T_0 \leq T$. Hence we can extend it in the interval $[0, T]$ with the fact that there exists $w \in \mathcal{U}$ satisfying $f(x_f, w) = 0$, leading to the fact that $\Gamma \neq \emptyset$. Suppose we have a sequence of trajectories $\{\gamma_i\}_{i=1}^{\infty} \subset \Gamma$ and correspondingly there exists a sequence of controls $\{u_i\}_{i=1}^{\infty} \subset \mathcal{U}^{[0,T)}$ such that $\dot{\gamma}_i = f(\gamma, u_i)$. Under $d_{\mathcal{U}}$, it is not hard to see that $\mathcal{U}^{[0,T)}$ is compact therefore there exists a subsequence $\{u_{i_j}\}_{j=1}^{\infty}$ admitting a limit

$$u = \lim_{j \to \infty} u_{i_j}$$

and $u \in \mathcal{U}^{[0,T)}$, with which, we have γ such that $\dot{\gamma} = f(\gamma, u)$. By simple adjust the proof of Theorem 1 in [54] from standard Euclidean space to a general compact manifold, we can have

$$\lim_{j \to \infty} d_{\Gamma}(\gamma, \gamma_{i_j}) = 0$$

from which it is obvious that $\gamma(0) = x_0$, $\gamma(T) = x_f$ and $\gamma \in (\Omega \backslash \mathcal{O})$ since $\Omega \backslash \mathcal{O}$ is compact. Thus, $\gamma \in \Gamma$, which implies that Γ is compact. □

Lemma 4.5.2. *Assume that (4.1) is true, there must exist a feasible path $\gamma \in \Gamma$ satisfying*

$$\left(\cup_{t \in [0,T]} B(\gamma(t), L)\right) \cap \mathcal{O} = \emptyset,$$

where $\mathcal{O}$ is the constrained set.

Proof. Because of (4.1), there exists a sequence of paths $\{\gamma_n\} \subset \Gamma$ satisfying

$$\lim_{n \to \infty} \inf_{t \in [0,T]} \sup_{r \in [0,\infty)} \{r : B(\gamma_n(t), r) \cap \mathcal{O} = \emptyset\} = L.$$

From Lemma 4.5.1, we know that Γ is compact, therefore $\lim_{n \to \infty} \gamma_n = \gamma_0 \in \Gamma$. For an arbitrary $t \in [0, T]$, denote

$$L(t) = \sup_{r \in [0,\infty)} \{r : B(\gamma_0(t), r) \cap \mathcal{O} = \emptyset\}$$

As the whole space Ω is compact, the limit $L(t) < \infty$. Since $\inf_{t\in[0,T]} L(t) = L$, one has $L \leq L(t)$ for arbitrary time $t \in [0, T]$. That is, fix the curve γ_0 defined as above, for all $t \in [0, T]$, $B(\gamma_0(t), L) \cap \mathcal{O} = \emptyset$ and thus

$$\left(\cup_{t\in[0,T]} B(\gamma_0(t), L)\right) \cap \mathcal{O} = \emptyset,$$

which proves the lemma. □

In the next few lemmas, we prove several results that ensure the generating graph algorithm (Algorithm 5 and 6) creating new points in feasible region when the radius l is small enough compared to L, and the process does not stop until reaching a neighborhood of the target state.

Lemma 4.5.3. *Given a continuous path γ and a closed set $\mathcal{V} \subset \Omega$ with $\gamma(0) = x, \gamma(T) = y$, and $x \in \mathcal{V}$ and $y \notin \mathcal{V}$, then there exists $z \in \gamma$ such that $z \in \partial\mathcal{V}$ and $\gamma \cap \partial\mathcal{V}$ is closed.*

Proof. The proof of this Lemma is the same as Lemma 3.6.4. □

Lemma 4.5.4. *Define $B_{\hat{d}}(x, r) = \{y : \hat{d}(x, y) < r\}$, then $B_{\hat{d}}(x, r) = B(x, r)$ if $(\Omega, \hat{d})$ is compact. And given $p \in \Omega$, $q \in S(p, L)$ and an orthonormal basis $N = \{e_i\}_{i=1}^d$ in $T_p\Omega$, then there exists $\exp(p, w, l) \in B(q, L)$ for some $w \in \{\pm e_i\}_{i=1}^d$ if Ω is a Lie group with Lie bracket vanishing and*

$$l < \frac{2L}{\sqrt{\lambda d}}$$

where d is the dimension of Ω and λ is the largest eigenvalue of $\hat{g}$.

Proof. Since $(\Omega, \hat{d})$ is compact, it is geodesic complete. If $y \in B(x, r)$, there exists a path γ linking x and y with $\dot{\gamma} = f(x, u)$ for some u and $\mathcal{L}(\gamma) < r$. On the other hand, we have a geodesic ζ linking x and y which has the property that $\mathcal{L}(\zeta) \leq \mathcal{L}(\gamma) < r$. Therefore, $y \in B_{\hat{d}}(x, r)$. Meanwhile, if $y \in B_{\hat{d}}(x, r)$, we can find a geodesic ζ linking x and y with $\mathcal{L}(\zeta) < r$. Since $\zeta \in \mathcal{S}(x, y)$, we have $y \in B(x, r)$. Thus, $B(x, r) = B_{\hat{d}}(x, r)$.

For the second part, without loss of generality, we assume that $\hat{g}(u,v) = u^T A u$ where $A = diag(\lambda_1, \cdots, \lambda_d)$ and $e_i = (0, \cdots, 1, \cdots, 0)$ with the i-th position to be 1. Let $\exp_x(u)$ be the exponential map (i.e.$\exp_x(u) = \gamma(1)$ where γ is the geodesic with $\gamma(0) = x, \dot{\gamma}(0) = u$), since Ω is flat and a Lie group, by Baker–Campbell–Hausdorff formula [55] we can see that $\exp_p(u+v) = \exp_{\exp_p(u)}(h(v))$ where $u, v \in T_p\Omega$ and $h(\cdot)$ is the parallel transport along the geodesic γ with $\gamma(0) = p, \dot{\gamma}(0) = u$ and h is an identity map under normal coordinates, thus is an diffeomorphism.

We consider the tangent space $T_p\Omega$. For arbitrary $z \in B(0_p, L)$, then

$$\hat{g}(z,z) = \sum_{i=1}^{d} \lambda_i z_i^2 = L^2$$

and there must be some k such that $\lambda_k z_k^2 \geq \frac{L^2}{d}$. Then we choose $w = sign(z_k)e_k$ to get

$$\begin{aligned}
\hat{g}(z - lw, z - lw) &= \sum_{i \neq k} \lambda_i z_i^2 + \lambda_k (z_k - sign(z_k) l)^2 \\
&< L^2 + \lambda_k l^2 - 2l|z_k|\lambda_k \\
&\leq \lambda_k l^2 - \frac{2L}{\sqrt{d}} \sqrt{\lambda_k} l + L^2.
\end{aligned}$$

$0 < l < \frac{2L}{\sqrt{\lambda d}}$ results in $0 < l < \frac{2L}{\sqrt{\lambda_k d}}$, which implies that $\hat{g}(z - lw, z - lw) < L$. Thus, $z \in B(lw, L)$ and

$$B(0, L) \subset \bigcup_{w \in \{\pm e_i\}} B(lw, l).$$

Next, since $(\Omega, \hat{g})$ is geodesic complete, we know that $exp_p(B(0,r)) = B(p,r)$, and it is

easy to see that

$$\begin{aligned}
\exp_p(B(lw, L)) &= \left\{\exp_p(lw + u) : u \in T_p\Omega, \hat{g}(u, u) < L\right\}, \\
B(\exp(p, w, l), L) &= \exp_{\exp(p,w,l)}(B(0_{\exp(p,w,l)}, L)) \\
&= \left\{\exp_{\exp(p,w,l)}(u) : u \in T_{\exp(p,w,l)}\Omega, \hat{g}(u, u) < L\right\} \\
&= \left\{\exp_{\exp(p,w,l)}(h(u)) : u \in T_p\Omega, \hat{g}(u, u) < L\right\}.
\end{aligned}$$

As for each $u \in T_p\Omega$ with $\hat{g}(u, u) < L$, $\exp_p(lw + u) = \exp_{\exp(p,w,l)}(h(u))$, we can conclude that $\exp_p(B(lw, L)) = B(\exp(p, w, l), L)$. Finally we have,

$$B(p, L) \subset \bigcup_{w\in\{\pm e_i\}} B(\exp(p, w, l), L)$$

which implies that

$$\bar{B}(p, L) \subset \bigcup_{w\in\{\pm e_i\}} \bar{B}(\exp(p, w, l), L)$$

where $\bar{A}$ is the closure of the set A. Thus, the conclusion in the lemma holds. □

Lemma 4.5.5. *Assume that* (4.1) *holds. Graph* $G = (V, E, K, p)$ *is generated by Algorithm 5 with* (4.2) *(or is generated by Algorithm 6 with* (4.4) *holding or* $(\Omega, \hat{d})$ *is a flat Lie group with the generating radius satisfying* $0 < l < \frac{2L}{\sqrt{\lambda d}}$ *and* $(\Omega, \hat{d})$ *being compact). If* $x_t \notin V$*, then the graph generating step of Algorithm 5 (Algorithm 6) does not stop, and there exists at least one point in the feasible region that can be added to* G *by the algorithm.*

Proof. Let us assume that the graph generating algorithm terminates after finite steps, returning a connected graph $G = (V, E)$ containing x_s. We denote

$$C = \cup_{v\in V}\bar{B}(v, L),$$

which is a closed set with $x_s \in C$. First we want to prove $x_t \in C$ by contradiction. Let us assume $x_t \notin C$. Take the path γ in Lemma 4.5.2, it is true that $\gamma(0) = x_s \in C$ while

$\gamma(T_0) = x_t \notin C$. Since γ is continuous, there exists at least one point that γ intersects with ∂C by Lemma 4.5.4, and we denote it as $\{\gamma(t_i)\}$. Let $x = \sup_{t_i} \gamma(t_i)$ be the last intersection point along the path. By Algorithm 6, we can find a vertex $v_c \in V$ such that $x \in S(v_c, L)$, which is the sphere centered at v_c with radius L. Since $x \in \gamma$ and by (4.1), we know that $B(x, L) \subset (\Omega \backslash \mathcal{O})$. Further we claim that there is no $v \in V \cap B(x, L)$, otherwise, $x \in B(v, L)$ implies $x \in C^o$, which is a contradiction with $x \in \partial C$.

Since the algorithm stopped, all current vertices $v \in V$ must have been tried to generate points around them. When the vertex v_c is chosen, by (4.2) for Algorithm 5 ((4.4) for Algorithm 6 or Lemma 4.5.4 when $(\Omega, \hat{d})$ is a flat Lie group), at least one point p can be generated in $B(x, L)$, which means either the algorithm should not terminate without having p, or p already exists before, which contradicts there is no vertex of G in $B(x, L)$. Therefore, we conclude that $x_t \in C$ if the algorithm stops.

If $x_t \in C$, since $x_t = \gamma(T_0)$, $B(x_t, L) \subset (\Omega \backslash \mathcal{O})$, then there is at least one vertex $v \in B(x_t, L)$, by the algorithm, an edge between x_t and v should be added to the graph G. Thus, if the algorithm stops, x_t and x_s are in the same connected component in the graph G. □

Now, we are ready to give the proof of the main theorems 4.2.1, 4.2.2 and 4.2.3, which can follow the same proof in 3.6.

Proof. (Proof of Theorem 4.2.1 and 4.2.2) Since Ω is compact and any two nodes have a distance of at least ϵ, the graph G is generated by Algorithm 5 or Algorithm 6 must contain finite number of vertices. Otherwise there is a cluster point in the vertex set, which implies that there exists two vertices in G whose distance is strictly smaller than the generating radius l regardless how small l is, which leads to a contradiction. Meanwhile, from the construction mechanism, G is always connected. Then Lemma 4.5.5 if Algorithm 6 is used (or (4.2) if Algorithm 5 is used) implies that G connects x_s and x_t. □

Proof. (Proof of Theorem 4.2.3) The proof of this theorem is the same as Theorem 3.2.2.

□

4.6 Conclusion

In this chapter, we generalize the method in Chapter 3 to solve a general control problem with part of the constraints unknown. In the algorithm, there are three major components: Graph Generation, Path Finding and Trajectory extension. And the key part is the Graph Generation, which is guided by the evolution of FPE. The algorithm guarantees success if the adding nodes strategy satisfies (4.2). Moreover, if the control system is strongly reversible, we provide a way using geodesics to construct the graph by solving initial value problems in each iteration especially when there exists directions that geodesics can be calculated trivially, the computation can be simplified further. Meanwhile, the to be visited nodes set on the tree, constructed by our algorithm strategy without stopping when target state is in the graph, is bounded by FPE.

We apply this method to several different control systems to illustrate the mechanism of our algorithm. The results are consistent with the properties we mentioned in this chapter and show the ability to handle high dimensional problems.

CHAPTER 5
METHOD OF EVOLVING JUNCTION ON OPTIMAL PATH PLANNING IN FLOW FIELDS

5.1 Introduction

Autonomous Underwater Vehicles (AUVs) are a class of submerged marine robots that are able to perform persistent missions in the ocean. Over the last few decades, AUVs have been widely applied to various applications, including ocean sampling [56, 57], surveillance and inspection [58], and many more. Since most of the applications require the AUVs to execute long-term missions in unknown and dynamic oceanic environments with minimum human supervisions, their success is highly dependent on the level of autonomy that the AUVs can achieve.

For robots operating in complex and dynamic environments, path planning is one of the crucial and fundamental functions to achieve autonomy. It aims to find a feasible or optimal path, under the influence of a dynamic flow field, for an AUV to reach a predefined goal point. Path planning has been studied extensively in robotics over the years. Several popular algorithms that have been applied in underwater vehicle navigation include graph based methods such as the A* method [59, 60], probability based methods like the Rapidly exploring Random Trees (RRTs) [61, 62], and methods that approximate the solution of HJ (Hamilton-Jacobi) equations, such as the Level Set Method (LSM) [63, 64].

When the A* method is applied for an AUV, the continuous flow field is partitioned into small cells. At each step, it compares the cost of going from the current position to its neighboring cells so as to identify a path with the lowest cost. However, when the resolution, which is inverse proportional to the cell size, is not high enough, it may fail to find a feasible path even if there exists one. Besides, the optimal path computed in the discretized

domain might not be optimal in the continuous field. To address the difficulties, Soulignac [65] introduces sliders to the propagation, which can move on the partitioned region boundaries. The sliders position during wavefront propagation is derived by continuous optimization, and then the optimal path is computed by the backtracking of wavefronts. RRT and RRT* explore the flow field by using random sampling, with a bias towards the unexplored area [15, 17]. Like many other probabilistic based methods, RRTs provide, guaranteed in the asymptotic sense, a globally optimal solution only when the samples are dense enough. LSM computes a propagating front, incorporating both flow and vehicle speeds, to approximate the solution of the HJ equation. Then the optimal path is computed by back tracking from the destination position. LSM can plan a shortest time path over time-varying flow field, usually at the cost of longer computational time.

Along with the advancements in computation power and sensor technology equipped in AUVs, there is a growing trend on deploying multiple AUVs to perform the adaptive environmental sampling and sensing tasks together [66, 67, 68]. With the field estimation information being shared among all agents, the mobile sensor network can exploit their mobility to perform autonomous missions such as gradient climbing and feature tracking in an uncertain environment [69]. However, the amount of information that can be shared among vehicles is limited due to constraints in communication capabilities. This demands information reduction before its transmission. For example, the reduced flow maps, not the actual flow fields, are often shared [70]. Analogous to the ground robot case, where traversibility analysis was previously applied to evaluate key features including the height, slope and roughness of terrains to determine its accessibility for ground robots [71, 72], the averaged flow and the spatial variation of the flow field are of the most critical patterns of the flow field that need to be shared by the mobile network in order to perform path planning for the agents. The averaged flow characterizes the averaged influence of flow field over vehicle dynamics in some spatial region, while the spatial variation of the flow field describes how the flow field can be partitioned into different regions. The flow field

partition and the averaged flow field over each region preserves the key patterns of the flow field, and path planning can thus be performed given these information.

In this chapter, we consider the path planning problem of a vehicle traveling in a d dimensional space. A novel method following the idea of Method of Evolving Junctions [2] is presented to solve this problem. We first divide the flow field into a piece-wise constant one and the total flow space is separated into different regions, within which the flow field is a constant vector. The vehicle moves across several regions to reach the destination. In each region, we can get the optimal solution for the sub-problem if the entering and leaving points are given. Following the Bellman principle, determining the optimal path is equivalent to finding the optimal entering and leaving points for each region. In this way, we reduce the original path planning in continuous flow field to a finite dimensional optimization with the variables being the intersection points between the path and the boundaries of regions. We call those points junctions. The resulting optimization problem in general is not convex, so we solve it by intermittent diffusion method [1]. The new algorithm has the following features:

1. Based on the divided flow regions, if the objective function is given as the total traveling time or the total energy consumption (often modeled by a quadratic function), we prove the structure of the global optimal path is a piece-wise constant velocity motion. Furthermore, our algorithm can find a global minimizer with probability 1 if some mild assumptions on the flow maps are provided.

2. It converts the infinite dimensional optimal control problem into a finite dimensional optimization problem, which can be solved fast.

3. Besides the flow field partition, the algorithm does not depend on any spatial discretizations, like the ones used in LSM. The computation accuracy is not limited by the mesh size used in the discretizations, and the computation complexity does not grow exponentially when the dimension increases.

In the next section, we present the formulation of the problem in the optimal control framework and the assumptions used in the chapter. In Section 5.3, we show how to transform the original problem into the finite dimensional optimization, and provide the algorithm. In Section 5.4, the detailed proof of the completeness of the algorithm is offered, accompanied with several numerical experiments in Section 5.5. At last, we end our paper with a brief conclusion.

5.2 Problem Statement

We consider the vehicle moving in the space $\Omega \subset \mathbb{R}^d$ with dimension d, equipped with a dynamic $\dot{x} = u + v$, where u is the environment flow velocity, and v is the vehicle velocity or the control variable, satisfying $v \in \mathcal{U}$, providing that $\mathcal{U}$ is a compact space such that $\|v\| \leq V$. We further assume that the flow field is divided into finite number of convex regions $\{R_\alpha\}_{\alpha \in I_R}$ by boundary curves (surfaces if is in $\mathbb{R}^d, d \geq 3$) $\{f_{\alpha\beta}\}_{(\alpha,\beta)\in I}$. Here

$$I = \{(\alpha, \beta) \in I_R \times I_R : dim(\partial R_\alpha \cap \partial R_\beta) = d - 1\},$$

where $dim(S)$ returns the dimension of the set S and ∂S is the boundary of S, and $f_{\alpha\beta}(x) = 0$ is the $(d-1)$-dimensional compact boundary of the region R_α and R_β (can be denoted as ∂R_α and ∂R_β respectively), on which there exists a piece-wise diffeomorphism

$$x_{\alpha\beta}(\lambda) : D \subset \mathbb{R}^{d-1} \longrightarrow \{y : f_{\alpha\beta}(y) = 0\}.$$

Also within each region, we suppose that the flow velocity u is a constant vector. Hence we can denote the flow velocity in each region R_α separately by u_α. The vehicle needs to be controlled from an initial position x_0, crossing different regions and finally reaches the target position x_f.

Since there could be infinitely many feasible paths linking x_0 and x_f, a cost function is introduced to measure the travel expense with respect to different potential trajectories.

We denote the cost function to be

$$J(v,T) = \int_0^T L(v(t))dt, \tag{5.1}$$

letting $\gamma(t)$ be a continuous path with $\gamma(0) = x_0$, $\gamma(T) = x_f$ and $\dot{\gamma}(t) = u + v$. In this paper, we study the problem with the cost function specifically being the total travel time, that is $L(x,v) = 1$, and the kinetic energy combining a constant running cost, in which case $L(x,v) = \|v\|^2 + C$ where C and $\|v\|$ are not simultaneously 0 for all $x \in \Omega$ because of technical issues which is discussed in Section 5.4.2. Our goal is to find the optimal control function $v(t)$ with minimum cost. The problem can be expressed as:

$$\begin{aligned} &\min_{v,T} \int_0^T L(v)dt \\ s.t.\quad &\dot{x} = v + u, \\ &x(0) = x_0, \\ &x(T) = x_f, \\ &\max_{t\in[0,T]} \|v\| \le V. \end{aligned} \tag{5.2}$$

In the next section, we will concretely discuss our method, the idea of which is to seek a way to change the infinite dimensional optimal control problem into a finite dimensional optimization problem.

5.3 Our Method

All possible trajectories will continuously pass through a sequence of regions, denote as $\{R_i\}_{i=1}^n$ in chronological order. Since the flow velocity is a constant vector in each region R_i, in each region, we have the following theorem:

Theorem 5.3.1. *The constant velocity motion is the optimal solution for*

$$\begin{aligned} \min_{v,T} & \int_0^T L(v)dt \\ s.t. \quad & \dot{x} = f(v+u), \\ & x(0) = x_0, \\ & x(T) = x_f, \\ & \max_{t\in[0,T]} \|v\| \leq V. \end{aligned}$$

with u a constant vector.

With Bellman's principle, it is the fact that the optimal path should be formed via a piece-wise constant velocity motion. Thus, we restrict the velocity of the vehicle in R_i to be a constant. Further, the regions are convex, therefore, the straight line path always lies in the certain region and therefore is continuous. Now we can introduce the notation v_i to be the vehicle velocity in R_i and the junction set $\{x_i\}_{i=0}^n$ where x_i are on the boundaries of different regions with x_0 the initial and $x_n = x_f$. In addition, without loss of generality, we assign x_0 to be the origin accompanied with x_f located on the y-axis. With these notations, we present our algorithm for different cost functions as below:

5.3.1 Minimize total travel time

When minimizing total travel time is the goal, we further let the vehicle move in maximum speed V. Then the cost function can be converted in the format below:

$$J(v,T) = \int_0^T ds = T = \sum_{i=1}^n t_i,$$

where t_i is the travel time in region R_i and is expressed as:

$$t_i = \frac{\|x_i - x_{i-1}\|}{\|v_i + u_i\|}.$$

Since flow and vehicle velocities are constant in R_i, the motion must be in straight line, which leads to

$$\begin{aligned}
&\frac{x_i - x_{i-1}}{\|x_i - x_{i-1}\|} = \frac{v_i + u_i}{\|v_i + u_i\|} \\
\Longrightarrow &\|v_i + u_i\|^2 - \frac{2(x_i - x_{i-1})^T u_i}{\|v_i + u_i\|} + \|u_i\|^2 - V^2 = 0 \\
\Longrightarrow &\|v_i + u_i\| = \frac{(x_i - x_{i-1})^T u_i}{\|x_i - x_{i-1}\|} \pm \left(\left(\frac{(x_i - x_{i-1})^T u_i}{\|x_i - x_{i-1}\|} \right)^2 + V^2 - \|u_i\|^2 \right)^{\frac{1}{2}}.
\end{aligned}$$

To minimize the travel time, we take the plus sign so that

$$\|v_i + u_i\| = \frac{(x_i - x_{i-1})^T u_i}{\|x_i - x_{i-1}\|} + \left(\left(\frac{(x_i - x_{i-1})^T u_i}{\|x_i - x_{i-1}\|} \right)^2 + V^2 - \|u_i\|^2 \right)^{\frac{1}{2}} \tag{5.3}$$

and have

$$\begin{aligned}
&J(v, T) = J(x_1, \cdots, x_{n-1}) \\
&= \sum_{i=1}^{n} \frac{1}{\|u_i\|^2 - V^2} \left((x_i - x_{i-1})^T u_i - \sqrt{((x_i - x_{i-1})^T u_i)^2 + \|x_i - x_{i-1}\|^2 (V^2 - \|u_i\|^2)} \right).
\end{aligned}$$

At last, since x_i is on the boundary of the regions R_1 and R_2, we have the smooth parameterization of each x_i as $x_i = x_i(\lambda_i)$ where $\lambda_i \in D \subset \mathbb{R}^{d-1}$, transforming finally the cost function to be

$$J(v, T) = J(\lambda_1, \cdots, \lambda_{n-1}) = J(x_1(\lambda_1), \cdots, x_{n-1}(\lambda_{n-1}))$$

and the problem is changed to a finite dimensional optimization problem

$$\min_{\lambda_i \in D, i=1,\cdots,n-1} J(\lambda_1, \cdots, \lambda_{n-1}) \tag{5.4}$$

with V, u_i, x_0 and $x_n = x_f$ given in advance.

5.3.2 Minimize energy

If the cost function is the kinetic energy with a constant running cost $C \geq 0$

$$J(v,T) = \int_0^T \|v\|^2 + C dt,$$

we first consider the constant velocity motion in each region R_i within time t_i. If the maximum vehicle speed V is large enough, we set

$$v = \frac{x_i - x_{i-1}}{t_i} - u_i$$

where

$$\|v_i\|^2 = \frac{\|x_i - x_{i-1}\|^2}{t_i^2} + \|u_i\|^2 - \frac{2(x_i - x_{i-1})^T u_i}{t_i} \leq V^2 \tag{5.5}$$

and hence if we fix the entrance and exit junctions x_i and x_{i-1}, the cost function in the specific region R_i can be rewritten into

$$\begin{aligned} J(t_i) &= \int_0^{t_i} \|v_i\|^2 + C ds \\ &= \frac{\|x_i - x_{i-1}\|^2}{t_i} + (\|u_i\|^2 + C)t_i - 2(x_i - x_{i-1})^T u_i \\ &\geq 2\sqrt{\|u_i\|^2 + C}\|x_i - x_{i-1}\| - 2(x_i - x_{i-1})^T u_i \end{aligned}$$

with equality holds at

$$t_i^* = \frac{\|x_i - x_{i-1}\|}{\sqrt{\|u_i\|^2 + C}}.$$

Hence from the above calculation, if

$$\|v_i\|^2 = 2\|u_i\|^2 + C - \frac{2(x_i - x_{i-1})^T u_i}{\|x_i - x_{i-1}\|}\sqrt{\|u_i\|^2 + C} \leq V^2, \tag{5.6}$$

we let the vehicle move in the speed of $\|v_i\|$ in (5.6). However, if the (5.6) does not hold,

we set the vehicle speed to be the maximum speed V and the cost function is reduced to be $J = (V^2 + C)t_i^*$ where

$$t_i^* = \frac{(x_i - x_{i-1})^T u_i - \sqrt{((x_i - x_{i-1})^T u_i)^2 + \|x_i - x_{i-1}\|^2 (V^2 - \|u_i\|^2)}}{\|u_i\|^2 - V^2}, \tag{5.7}$$

which is a root of

$$(\|u_i\|^2 - V^2)t_i^2 - 2(x_i - x_{i-1})^T u_i t_i + \|x_i - x_{i-1}\|^2 = 0.$$

Thus the problem becomes

$$J(v, T) = J(x_1, \cdots, x_{n-1}) = \sum_{i=1}^{n} f_i(x_i, x_{i-1}),$$

where

$$f_i(x_i, x_{i-1}) = \begin{cases} 2\sqrt{\|u_i\|^2 + C}\|x_i - x_{i-1}\| - 2(x_i - x_{i-1})^T u_i & \text{if (5.6) holds} \\ (V^2 + C)t_i^* & \text{otherwise} \end{cases}$$

where t_i^* is as in (5.7). By using the same parametrization, we finally have the problem to be a finite dimensional optimization formulated as

$$\min_{\lambda_i \in [0,1]^{d-1}, i=1,\cdots,n-1} L(\lambda_1, \cdots, \lambda_{n-1}) \tag{5.8}$$

where

$$L(\lambda_1, \cdots, \lambda_{n-1}) = f_1(x_1(\lambda_1), x_0) + f_n(x_{n-1}(\lambda_{n-1}), x_f) + \sum_{i=2}^{n-1} f_i(x_i(\lambda_i), x_{i-1}(\lambda_{i-1})).$$

Remark 4. *The objective function is well defined resulting from the fact that $t_i^* > 0$ if there exists a feasible trajectory from x_{i-1} to x_i in R_i and $V \neq \|u_i\|$. This can be proven as*

below:

If $(x_i - x_{i-1})^T u_i \leq 0$*, unless* $V > \|u_i\|$*, there does not exists a feasible path. Therefore,*

$$(x_i - x_{i-1})^T u_i < \sqrt{((x_i - x_{i-1})^T u_i)^2 + \|x_i - x_{i-1}\|^2 (V^2 - \|u_i\|^2)}.$$

Since $\|u_i\|^2 - V^2 < 0$*, we have* $t_i^* > 0$*. On the other hand, if* $(x_i - x_{i-1})^T u_i > 0$*, we can have two cases:* $V > \|u_i\|$*, which shares the same conclusion as the first case, and* $V < \|u_i\|$*. In the latter circumstance, since* $\|u_i\|^2 - V^2 > 0$ *and*

$$(x_i - x_{i-1})^T u_i > \sqrt{((x_i - x_{i-1})^T u_i)^2 + \|x_i - x_{i-1}\|^2 (V^2 - \|u_i\|^2)},$$

it is still true that $t_i^* > 0$*.*

Meanwhile, When $(x_i - x_{i-1})^T u_i > 0$*,* $t_i^* > 0$ *still holds if* $V = \|u_i\|$ *and actually*

$$\begin{aligned} t_i^* &= \lim_{V^2 - \|u_i\|^2 \to 0} \frac{(x_i - x_{i-1})^T u_i - \sqrt{((x_i - x_{i-1})^T u_i)^2 + \|x_i - x_{i-1}\|^2 (V^2 - \|u_i\|^2)}}{\|u_i\|^2 - V^2} \\ &= \lim_{V^2 - \|u_i\|^2 \to 0} \frac{\|x_i - x_{i-1}\|^2}{(x_i - x_{i-1})^T u_i + \sqrt{((x_i - x_{i-1})^T u_i)^2 + \|x_i - x_{i-1}\|^2 (V^2 - \|u_i\|^2)}} \\ &= \frac{\|x_i - x_{i-1}\|^2}{2(x_i - x_{i-1})^T u_i} > 0. \end{aligned}$$

However, if $(x_i - x_{i-1})^T u_i \leq 0$*,* $V = \|u_i\|$ *becomes a singular point since there is no feasible path. Thus, in this case, we cannot formally solve the problem.*

Furthermore, f_i and t_i^* have the following properties:

Proposition 6. $f_i(x_i, x_{i-1})$ *is differentiable.*

We will give the proof of this property later in Section 5.4. And with this proposition, we can take the derivative of the objective function, which is pivotal for us to use the intermittent diffusion method to handle this problem.

5.3.3 Intermittent Diffusion

To solve (5.4) and (5.8), which are both differentiable, we use the intermittent diffusion (ID) to get the global minimizer [1], the key idea of which is adding white noise to the gradient flow intermittently. Namely, we solve the following stochastic differential equation (SDE) on the configuration space

$$d\lambda = -\nabla J(\lambda)d\theta + \sigma(\theta)dW(\theta) \tag{5.9}$$

where $\lambda = (\lambda_1, \cdots, \lambda_{n-1}) \in D^{n-1}$ and $W(\theta)$ is the standard Brownian motion. The diffusion is a piece-wise constant function defined by

$$\sigma(\theta) = \sum_{i=1}^{N} \sigma_i I_{[S_i,T_i]}(\theta)$$

where σ_i are constant and $I_{[S_i,T_i]}(\theta)$ is the characteristic function on interval $[S_i, T_i]$ with $0 \leq S_1 < T_1 < S_2 < T_2 < \cdots < S_N < T_N < S_{N+1} = T$.

Thus, if $\sigma(\theta) = 0$, we obtain the gradient flow back while when $\sigma(\theta) \neq 0$, the solution of (5.9) has positive probability to escape the current local minimizer. The theory of ID indicates that the solution of (5.9) visits the global minimizer of J with probability arbitrarily close to 1 if $\min_i |T_i - S_i|$ is large enough, which is guaranteed by the following theorem in [1]:

Theorem 5.3.2. *Let Q be the set of global minimizers, U be a small neighborhood of Q and λ_{opt} the optimal solution obtained by the ID process. Then for any given $\epsilon > 0$, there exists $\tau > 0$, $\sigma_0 > 0$ and $N_0 > 0$ such that if $T_i - S_i > \tau$, $\sigma_i < \sigma_0$ (for $i = 1, \cdots, N$) and $N > N_0$,*

$$\mathbb{P}(x_{opt} \in U) \geq 1 - \epsilon.$$

We use the forward Euler discretization to discretize the above SDE and get

$$\lambda^{k+1} = \lambda^k - h\nabla J(\lambda^k) + \sigma_k \xi^k \sqrt{h}. \tag{5.10}$$

The constant h is the step size, σ_k is the coefficient chosen to add the intermittent perturbation and $\xi^k \sim \mathcal{N}(0,1)$ is a Gaussian random variable. In practice, the global minimizer can be reached by tuning the white noise strength σ_k as well as setting the total evolution round N long enough.

Adding and Deleting Junctions

We notice that the number of junctions may change during the evolution. Using the assumption that

Assumption 5.3.1. *There exists at least one optimal trajectory that goes through each region no more than once,*

we propose a heuristic way to handle the number variation of junctions. We denote the current junction set to be $\{x_i\}_{i=1}^K$ with $f_{\alpha_i\alpha_{i+1}}(x_i) = 0$, that is, x_i is on the boundary of R_{α_i} and $R_{\alpha_{i+1}}$. Without loss of generality, we assume R_{α_i} is the one the vehicle exits while $R_{\alpha_{i+1}}$ is the region entered. Thus, using the boundary function $f_{\alpha_i\alpha_{i+1}}$, we could form a chain (or a linked list)

$$f_{\alpha_1\alpha_2} \to f_{\alpha_2\alpha_3} \to \cdots \to f_{\alpha_K\alpha_{K+1}}, \tag{5.11}$$

where R_{α_1} contains the initial location x_0 and the target location x_f lies in the region $R_{\alpha_{K+1}}$. Also, we can write the corresponding junction chain as

$$x_1 \to x_2 \to \cdots \to x_K. \tag{5.12}$$

Now as the gradient flow evolves, it is inevitable to be in the situation where in the next evolution, $\lambda \notin D$, in which case, there must be some junctions moving out of the current

boundaries curves (or surfaces). In every step after the update via (5.10), we check the status of each junction point. Then we need to add or remove corresponding junctions. We assume further that a junction is at most adjacent to 3 different regions. We express this assumption as below:

Assumption 5.3.2. *Denote the set X to be*

$$X = \{x \in \Omega : \exists (\alpha_1, \beta_1), (\alpha_2, \beta_2) \in I, \text{ such that } f_{\alpha_1\beta_1} \not\equiv f_{\alpha_2\beta_2}, f_{\alpha_1\beta_1}(x) = 0, f_{\alpha_2\beta_2}(x) = 0\},$$

the set $F(x) = \{\{z : f_{\alpha\beta}(z) = 0\} : (\alpha, \beta) \in I, \}$ and $R(x) = \{R_\alpha : \alpha \in I_R, x \in \partial R_\alpha\}$. Now assume that for all $x \in X$, $R(x)$ contains 3 elements. (This assumption is to simplify the technical issue when numerically solving the problem. In general, if proper strategies are implemented, this assumption can be replaced.)

If in the $(k+1)$-th step of intermittent diffusion, junction x_i^{k+1} moves out of the current boundary, which originally stays on $f_{\alpha_i\alpha_{i+1}}$, that is, $f_{\alpha_i\alpha_{i+1}}(x_i^k) = 0$. Here, to distinct the junction x_i in different evolution steps, we let x_i^k be the junction x_i at step k while x_i^{k+1} the same junction but at step $k+1$.

We find the point $x \in \partial\{y : f_{\alpha_i\alpha_{i+1}}(x^k) = 0\}$, such that $x = \mu x_i^{k+1} + (1-\mu)x_i^k$ for some $\mu \in [0, 1]$. It is true that either $x \in \partial\Omega$ or $x \in X$ defined in Assumption 5.3.2. If it is the first case, we replace x_i in (5.12) with x and keep (5.11) the same.

However if $x \in X$, we need to modify the boundary and junction chain by added and deleting junctions. Again by Assumption 5.3.2, we have for small enough $\epsilon > 0$, there exists at most three regions that intersect with $B(x, \epsilon)$, which is the ball centered at x with radius ϵ. Denoting the regions near x as $\{R_{\alpha_i}, R_{\alpha_{i+1}}, R_\beta\}$, we replace $f_{\alpha_i\alpha_{i+1}}$ in (5.11) with $f_{\alpha_i\beta} \to f_{\beta\alpha_{i+1}}$ and the chain becomes

$$f_{\alpha_1\alpha_2} \to f_{\alpha_2\alpha_3} \to \cdots \to f_{\alpha_i\beta} \to f_{\beta\alpha_{i+1}} \cdots \to f_{\alpha_K\alpha_{K+1}}. \tag{5.13}$$

Meanwhile, the corresponding junction chain becomes (by replacing x_i with two new junctions on $f_{\alpha_i\beta}$ and $f_{\beta\alpha_{i+1}}$):

$$x_1 \to \cdots \to x_{i-1} \to y_1 \to y_2 \to x_{i+1} \to \cdots \to x_K.$$

Now we perform the same procedure on all junctions and finally get a modified chain. We denote the new boundary and junction chains with the notation of the original chain in (5.11) and (5.12).

Next, if there exists some $f_{\alpha_{i_1}\alpha_{i_1+1}}$ and $f_{\alpha_{i_2}\alpha_{i_2+1}}$ such that $\alpha_{i_1} = \alpha_{i_2+1}$, then we delete the junctions $x_{i_1}, x_{i_1+1} \cdots, x_{i_2}$ and get the new chain, with which, we have a new optimization problem and conduct (5.10) to get the optimal in this circumstance.

We use a simple example in a 2-D space shown in Fig. 5.1 to illustrate the whole procedure and explain the result. Initially as is in Fig. 5.1(a), the path links the start location $(0, 0)$ and the target $(0, 20)$, passing R_1, R_2, R_3 consecutively with $f_{12}(x_1) = f_{23}(x_2) = 0$. We suppose that in the next update with (5.10), x_1 moves out of $\partial R_1 \cap \partial R_2$ (i.e. $f_{12}(x_1^+) \neq 0$ with x_1^+ is updated by (5.10) via the renew of the parameters λ), which means that we need to replace f_{12} with a new chain going from R_1 to R_2 and passing R_3. To this end, we add two new junctions y_1, y_2, and the chain of boundaries changes from $f_{12} \to f_{23}$ to $f_{13} \to f_{32} \to f_{23}$ with the vehicle moves in R_2 on y_2x_2 segment (the path is depicted in Fig. 5.1(b)). From here, if we further suppose that x_2 will also be outside the boundary $\partial R_2 \cap \partial R_3$, we following the same procedure and add junctions y_3, y_4. The chain of boundaries becomes $f_{13} \to f_{32} \to f_{21} \to f_{13}$ with the new path shown in Fig. 5.1(c) and the junction chain begin $y_1 \to y_2 \to y_3 \to y_4$. Since f_{13} and f_{21} appears in the new chain, we delete those two boundaries, as well as all of them in between those two elements in the chain. After performing this step, we have the chain contains a single element f_{13}. Meanwhile, we delete all the corresponding junctions and have y_4 as the final junction. With this, we further perform the intermittent diffusion to do the calculation.

Remark 5. *We notice that when $f_{\alpha\beta}$ and $f_{\xi\alpha}$ appear in the chain of boundaries with $f_{\alpha\beta}$ ahead of the other one, it means that the path exits R_α at some time but returns to R_α later. By Assumption 5.3.1, we know that the path with cycle is never the minimizer of the problem, thus we can conduct path segment branch to re-form the path into the desired no-cycle format. Meanwhile, the whole procedure implicitly eliminate the junctions that potentially will locate at the same spot. As is in the example, while doing continuous gradient flow and restrict the junctions sticking on the current boundaries, x_1 and x_2 will finally overlap, in which case a junction reduction is needed (the same as in [1]). Our procedure in this example shares equivalent result and naturally combines add and delete junctions in a single step.*

Now we can summarize our algorithm as Algorithm 7 with the objective function being (5.4) or (5.8).

5.4 Completeness

In this section, we demonstrate that Algorithm 7 is complete if $L = 1$ or $L = \|v\|^2 + C$. We first offer the following theorem and provide detailed proof in the next two sub-sections 5.4.1 and 5.4.2.

Theorem 5.4.1. *If the flow field is piece-wise constant and*

$$\max_{\alpha \in I_R} \|u_\alpha\| < V. \tag{5.14}$$

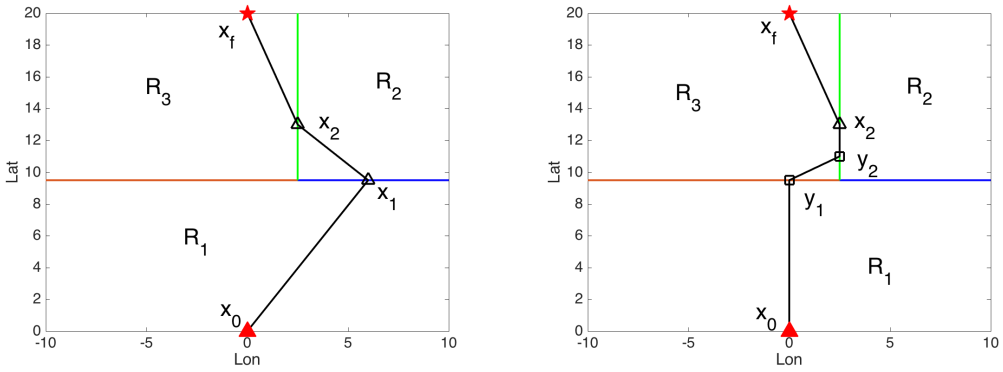
(5.3.1) *hold, let Q be the set of global minimizers, U be a neighborhood of Q. Then for any $\epsilon > 0$, there exists T_0, N_0, σ_0 such that if $T_i > T_0$, $\sigma_i < \sigma_0$ (for $i = 1, 2, \cdots, N$) and $N > N_0$ where T_i, σ_i, N are parameters in Algorithm 7, $\mathbb{P}(\gamma_{opt} \in U) \geq 1 - \epsilon$, where γ_{opt} is the optimal solution found by Algorithm 7. Thus, Algorithm 7 is complete.*

Algorithm 7: Shortest Path Find in Piece-wise Constant Flow Field

Data: flow field velocity $\{v_i\}$, boundaries $\{f_{\alpha\beta}\}$, glider maximum speed V, initial position x_0 and target position x_f

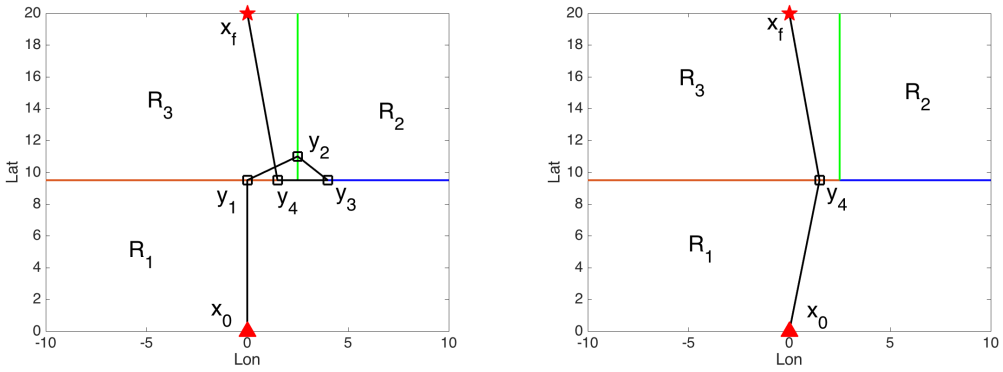
Output: the optimal trajectory γ_{opt}, junctions $\{x_i(\lambda_i)\}$

Set γ_0 be the straight line from x_0 to x_f;
Set junction set be the intersections between γ_0 and $\{f_{\alpha\beta}\}$;
Set evolution step number N;
Choose threshold ϵ;
for $i = 1, \cdots, N$ **do**
    $\gamma_i = \gamma_{i-1}$;
    Choose perturbation duration T_i;
    Choose perturbation intensity σ_i;
    for $j = 1, \cdots, T$ **do**
        Update γ_i using (5.10);
        Add and remove junctions for γ_i when necessary;
    end
    Set $\sigma_i = 0$;
    while *not converges* **do**
        Update γ_i using (5.10);
        Add and remove junctions for γ_i when necessary;
    end
end
Set $\gamma_{opt} = \arg\min_{i \leq N} J(\gamma_i)$;



(a) Initial: the path with x_1, x_2 being junctions that move out the corresponding boundaries in the next update ($f_{12} \to f_{23}$).

(b) First step: remove x_1 and insert y_1, y_2 to have the sub-path going from R_1 to R_2 and passing R_3 meanwhile ($f_{13} \to f_{32} \to f_{23}$).



(c) Second step: remove x_2 and insert y_3, y_4 to have the sub-path going from R_2 to R_3 and passing R_1 at the same time ($f_{13} \to f_{32} \to f_{21} \to f_{13}$).

(d) Last Step: remove y_1, y_2, y_3 and have the final path since we have f_{12} and f_{21} in the chain. Finally, the chain is f_{13}.

Figure 5.1: The whole add and eliminate junction procedure in a 3-region space.

The idea is that by Bellman principle, optimal trajectory admits a optimal sub-structure property, that is, any piece of the optimal trajectory is also optimal for the sub-problem. By applying this principle, we consider the path segment in each single region, and try to construct a solution ψ with two types of objective function described in Section 5.2, for the Hamilton-Jacobi-Bellman equation (HJB)

$$\psi_t(x,t) + H(x, \nabla\psi(x,t)) = 0 \tag{5.15}$$

where

$$H(x,p) = \max_{\|v\| \le V} \left\{ p^T (v+u) - L(x,v) \right\}.$$

is the Hamiltonian and

$$\psi(x,t) = \min_{v} \left\{ \int_0^t L(\gamma, v) ds : \dot{\gamma} = v + u, \gamma(0) = x_0, \gamma(t) = x, \max_{s \in [0,t]} v(x) \le V \right\}$$

is the value function. Since the original problem takes the minimum over all possible time, we take $\min_t \psi(x,t)$ to get the optimizer in the given region and claim that the corresponding motion gives a global optimal for the sub-problem in the single region.

5.4.1 Total Travel Time

$L(x,v) = 1$ for total travel time minimization. To construct the value function at the point (x,t), we introduce the maximum speed constant velocity motion in the region with flow velocity u, that is, in this region, the vehicle moves in straight line from x_0 to x with velocity $v+u$ and $\|v\| = V$, $\|v+u\|$ is given by (5.3). We claim that

Lemma 5.4.2. *In a constant flow field, the maximum speed straight line motion is optimal if we minimize the total travel time*

$$\begin{aligned} &\min_{v,T} \int_0^T dt \\ s.t. \quad &\dot{x} = v + u, \\ &x(0) = x_0, \\ &x(T) = x_f, \\ &\max_{t \in [0,T]} \|v\| \le V. \end{aligned}$$

Proof. We write the value function as

$$\psi(x,t) = \frac{\|x - x_0\|}{\|v + u\|}.$$

To make the problem complete, we define $\psi(x,t) = +\infty$ if the vehicle cannot reach x in time t, which gives the final value function to be

$$\psi(x,t) = \begin{cases} \frac{\|x-x_0\|}{\|v+u\|} & \frac{\|x-x_0\|}{\|v+u\|} \leq t \\ +\infty & \text{otherwise} \end{cases}.$$

If only the reachable part is considered, from the above equation, we can calculate $\psi_t = 0$ and

$$\nabla\psi = \frac{1}{\|v+u\|^2}\left(\|v+u\|\frac{x-x_0}{\|x-x_0\|} - \|x-x_0\|\nabla\|v+u\|\right), \tag{5.16}$$

to which end, first we can rewrite $v = v_0 + v^\perp$ and $V^2 = \|v_0\|^2 + \|v^\perp\|^2$ if we denote

$$\begin{aligned} v_0 &= \frac{(x-x_0)}{\|x-x_0\|}\frac{(x-x_0)^T v}{\|x-x_0\|}, \\ v^\perp &= \left(I - \frac{(x-x_0)}{\|x-x_0\|}\frac{(x-x_0)^T}{\|x-x_0\|}\right) v, \end{aligned}$$

where I is the identity matrix. And u can be decomposed in the same manner $u = u_0 + u^\perp$. Secondly, it is easy to see that $v^\perp = -u^\perp$ since $(v+u)/\|v+u\| = (x-x_0)/\|x-x_0\|$. Then, we see that $\|v+u\| = \|v_0\| + \|u_0\|$ and

$$\sqrt{\left(\frac{(x-x_0)^T u}{\|x-x_0\|}\right)^2 + V^2 - \|u\|^2} = \sqrt{\|u_0\|^2 + \|v_0\|^2 + \|v^\perp\|^2 - \|u_0\|^2 - \|u^\perp\|^2} = \|v_0\|.$$

Hence, we have

$$\begin{aligned}\nabla\|v+u\| &= \nabla\left(\frac{(x-x_0)^T u}{\|x-x_0\|}+\sqrt{\left(\frac{(x-x_0)^T u}{\|x-x_0\|}\right)^2+V^2-\|u\|^2}\right)\\ &= \frac{\|v+u\|}{\|v_0\|}\nabla\frac{(x-x_0)^T u}{\|x-x_0\|}\\ &= \frac{\|v+u\|}{\|x-x_0\|\|v_0\|}\left(I-\frac{(x-x_0)}{\|x-x_0\|}\frac{(x-x_0)^T}{\|x-x_0\|}\right)u\\ &= \frac{\|v+u\|}{\|x-x_0\|\|v_0\|}u^{\perp}.\end{aligned}$$

Taking $\nabla\|v+u\|$ back to (5.16) and noticing that $v^{\perp}=-u^{\perp}$, we reduce the gradient to be

$$\nabla\psi \;=\; \left(\frac{x-x_0}{\|x-x_0\|}-\frac{u^{\perp}}{\|v_0\|}\right)\frac{1}{\|v+u\|}=\frac{v}{\|v_0\|\|v+u\|}.$$

With the above equation, the Hamiltonian is

$$\begin{aligned}H &= \sup_{\hat{v}:\|\hat{v}\|\le V}\left(\nabla\psi^T(\hat{v}+u)-1\right)\\ &= \sup_{\hat{v}:\|\hat{v}\|\le V}\left\{\frac{v^T}{\|v_0\|}\frac{\hat{v}+u}{\|v+u\|}\right\}-1\\ &= \frac{1}{\|v+u\|}\left(\sup_{\hat{v}:\|\hat{v}\|\le V}\left\{\frac{v^T\hat{v}}{\|v_0\|}\right\}+\|u_0\|-\frac{\|u^{\perp}\|^2}{\|v_0\|}\right)-1\\ &= \frac{1}{\|v+u\|}\left(\frac{V^2}{\|v_0\|}+\|u_0\|-\frac{\|u^{\perp}\|^2}{\|v_0\|}\right)-1\\ &= \frac{1}{\|v+u\|}(\|v_0\|+\|u_0\|)-1=0,\end{aligned}$$

which leads to the conclusion that the value function induced by the maximum speed constant velocity motion solves the Hamilton-Jacobi equation, thus is the optimal moving pattern in a constant flow speed region since $\min_t\psi=\psi$. □

Meanwhile, using the same notation and logic, we can give the proof of Proposition 6:

Proof. (Proof of Proposition 6) Since $f_i(x_i,x_{i-1})=f_i(x_i-x_{i-1})$, we only need to consider

the differentibility of

$$g(a) = \begin{cases} g_1(a) & \text{if (5.6) holds} \\ g_2(a) & \text{otherwise} \end{cases}$$

where

$$\begin{aligned} g_1(a) &= 2\sqrt{\|u\|^2 + C}\|a\| - 2a^T u \\ &= 2\|a\| \left(\sqrt{\|u\|^2 + C} - \|u_0\| \right), \\ g_2(a) &= \frac{V^2 + C}{\|u\|^2 - V^2} \left(a^T u - \sqrt{(a^T u)^2 + \|a\|^2 (V^2 - \|u\|^2)} \right) \\ &= \frac{(V^2 + C)\|a\|}{\|u_0\| + \|v_0\|}. \end{aligned}$$

First of all, when equality in (5.6) holds, we have

$$\sqrt{\|u\|^2 + C} = \|u_0\| \pm \|v_0\| \Longrightarrow \sqrt{\|u\|^2 + C} = \|u_0\| + \|v_0\|. \tag{5.17}$$

We take the plus sign since $\sqrt{\|u\|^2 + C} \geq \|u\|$. Meanwhile from (5.6) and (5.17), we can derive the following equation

$$V^2 + C = 2\|v_0\|(\|u_0\| + \|v_0\|)$$

Therefore, $g(a)$ is continuous. By doing similar calculation previously in this section, we have

$$\begin{aligned} \nabla g_1 &= 2 \left((\sqrt{\|u\|^2 + C}) \frac{a}{\|a\|} - u \right) = v_0 - u^{\perp} = 2v, \\ \nabla g_2 &= \frac{V^2 + C}{\|v_0\|\|v + u\|} v = 2v, \end{aligned}$$

which gives us the desired result. □

5.4.2 Quadratic Energy with a constant running cost

In this case, $L(x, v) = \|v\|^2 + C$ where $C \geq 0$ is a constant running cost. To calculate the optimal solution for the vehicle running from x_0 to the target x in a constant flow velocity field, we again study the constant speed straight line motion. However in this circumstance, the vehicle may no longer travel with maximum speed, hence we take the travel time in the region into consideration. Suppose that the the vehicle moves from x_0 to x in time t, we set the vehicle velocity to be

$$v = \frac{x - x_0}{t} - u,$$

assuming that

$$\|v\|^2 = \frac{\|x - x_0\|^2}{t^2} + \|u\|^2 - \frac{2(x - x_0)^T u}{t} \leq V^2. \tag{5.18}$$

Then the value function is

$$\psi(x, t) = (\|v\|^2 + C)t = \frac{\|x - x_0\|^2}{t} - 2(x - x_0)^T u + (C + \|u\|^2)t.$$

Further we take

$$\psi(x, t) = \begin{cases} \frac{\|x-x_0\|^2}{t} - 2(x - x_0)^T u + (c + \|u\|^2)t & \|v\| \leq V \\ +\infty & \text{otherwise} \end{cases}.$$

Then by direct calculation with the finite part of ψ, we have

$$\psi_t = C + \|u\|^2 - \frac{\|x - x_0\|^2}{t^2}, \tag{5.19}$$

$$\nabla\psi = \frac{2(x - x_0)^T u}{t} - 2u, \tag{5.20}$$

$$\|\nabla\psi\|^2 = \frac{4\|x - x_0\|^2}{t^2} + 4\|u\|^2 - \frac{8(x - x_0)^T u}{t} \tag{5.21}$$

The Hamilton-Jacobi equation is in the form of

$$\psi_t + \sup_{v:\|v\|\leq V} \left\{\nabla\psi^T(v+u) - \|v\|^2 - C\right\} = 0. \tag{5.22}$$

To solve the optimization part of (5.22), we denote

$$F(v) = \nabla\psi^T(v+u) - \|v\|^2 - C$$

and calculate the critical point of it as

$$v^* = \frac{1}{2}\nabla\psi$$

which means that the optimal is

$$H = \sup_{v:\|v\|\leq V} F(v) = \frac{1}{4}\|\nabla\psi\|^2 + \nabla\psi^T u - C \tag{5.23}$$

and by (5.20), we have

$$\|v^*\|^2 = \frac{1}{4}\|\nabla\psi\|^2 = \frac{\|x-x_0\|^2}{t^2} + \|u\|^2 - \frac{2(x-x_0)^T u}{t} \leq V^2 \tag{5.24}$$

which leads to the fact that $F(v^*) = \sup_{v:\|v\|\leq V} F(v)$. Let us take (5.20,5.21) into (5.23) and the result is

$$H = \frac{\|x-x_0\|^2}{t^2} - \|u\|^2 - C. \tag{5.25}$$

Combining (5.19) and (5.25) finally results in the constructed ψ being the solution of (5.22).

Based on the solution ψ, we further find the minimizer over time t and solve the minimization problem as follow:

$$\min_{t\geq 0} \psi = \min_{t\geq 0}(\|v\|^2 + C)t = \frac{\|x-x_0\|^2}{t} - 2(x-x_0)^T u + (C + \|u\|^2)t.$$

It is easy to see that the global minimizer of ψ over t is

$$t^* = \frac{\|x - x_0\|}{\sqrt{C + \|u\|^2}}$$

and the corresponding minimum is

$$\psi^* = 2\|x - x_0\|\sqrt{C + \|u\|^2} - 2(x - x_0)^T u. \tag{5.26}$$

Thus, if t^* is reachable, that is, using (5.18), we have

$$\|v(t^*)\|^2 = C + 2\|u\|^2 - \sqrt{C + \|u\|^2}\frac{2(x - x_0)^T u}{\|x - x_0\|} \leq V^2$$

the optimal is given as (5.26).

On the other hand, if $\|v(t^*)\| > V$, the global minimizer t^* is on longer in the domain of our problem. In this case, we notice that $\|v\|^2$ is decreasing on the interval

$$\left[t^*, \frac{(x - x_0)^T u}{\|x - x_0\|^2}\right],$$

and is increasing on

$$\left[\frac{(x - x_0)^T u}{\|x - x_0\|^2}, +\infty\right).$$

Also by noticing that $\lim_{t\to+\infty} \|v\|^2 = \|u\|^2 \leq V^2$, we conclude that there exists $t_0 > t^*$ when $t \geq t_0 > t^*$, $\|v\| \leq V$. Meanwhile, when $t > t^*$, ψ is monotone increasing with respect to t. Hence, to get the minimum, we should take the time $t = t_0$, where $\|v(t_0)\| = V$. By taking the equality in (5.18), we have then

$$(\|u\|^2 - V^2)t^2 - 2(x - x_0)^T u t + \|x - x_0\|^2 = 0,$$

from which we have

$$t_0 = \frac{(x-x_0)^T u - \sqrt{\left((x-x_0)^T u\right)^2 + \|x-x_0\|^2(V^2 - \|u\|^2)}}{\|u\|^2 - V^2},$$

and $\min_t \psi = (V^2 + C)t_0$. Thus, we can conclude that

Lemma 5.4.3. *In a constant flow field, the minimizer of energy optimal problem*

$$\begin{aligned} &\min_{v,T} \int_0^T \|v\|^2 + C dt \\ s.t.\quad &\dot{x} = v + u, \\ &x(0) = x_0, \\ &x(T) = x_f, \\ &\max_{t\in[0,T]} \|v\| \le V. \end{aligned}$$

is the constant velocity motion in the speed of $\|v\|$, where

$$\|v\| = \begin{cases} \left(C + 2\|u\|^2 - \sqrt{C + \|u\|^2}\frac{2(x-x_0)^T u}{\|x-x_0\|}\right)^{1/2} & \textit{if (5.24) holds} \\ V & \textit{otherwise} \end{cases}$$

Remark 6. *When* (5.24) *does not hold, the minimizer of the energy optimal problem is the same as the minimizer of travel time optimal one. Hence, if the constant running cost C large enough, solving energy optimal problem is equivalent to solve travel time optimal problem.*

By the proof of Lemma 5.4.2 and Lemma 5.4.3, we can have the following theorem, which tell the optimal path structure within each constant flow region, given entrance and exit locations.

Theorem 5.4.4. *In each constant flow region, given the entrance and exit locations, the*

vehicle motion defined in Lemma 5.4.2 and Lemma 5.4.3 solves the HJB equation

$$\psi_t(x,t) + \max_v \{\nabla\psi(x,t)^T(u+v) - L(x,v)\} = 0$$

for $L = 1$ and $L = \|v\|^2 + C$ respectively. Moreover, among all the solutions of the above HJB, motions in these two lemmas gives the path with shortest time. Thus, we have the optimal solution of the sub-problem in each region.

Proof. (Proof of Theorem 5.4.1) We combine Theorem 5.4.4, together with Bellman principle, the global optimal path must be in the structure of constant motion within each flow region. To prove that the proposed algorithm is convergent, we only need to show that there exists a global minimizer $\lambda^* = (\lambda_1^*, \cdots, \lambda_k^*)$, around which there is a closed neighborhood $U \subset \prod_{i=1}^k D_i$ such that $vol(U) > 0$ (vol is the product Lebesgue measure in $\prod_{i=1}^k D_i$) and for all $\lambda \in U$, the gradient flow $\dot{\lambda} = -\nabla J(\lambda)$ converges to λ^*. If this condition holds, we can following the proof of intermittent diffusion and get the desired results.

To this end, if there exists such U that $vol(U) > 0$ and for all $\lambda \in U$, we have $J(\lambda) \leq J(\mu)$ for arbitrary $\mu \in S$ for some $S \subset U$, then the prove is done. Now if the global minimizers are isolated, then given any global minimizer $\lambda^* = (\lambda_1^*, \cdots, \lambda_k^*)$, since J is continuous differentiable, we can have a closed neighborhood $U \subset \prod_{i=1}^k D_i$ with $vol(U) > 0$ (within the neighborhood, the dimension of the domain does not change) such that $J(\lambda) > J(\lambda^*)$ and $\nabla J(\lambda) \neq 0$ for all $\lambda \in U \backslash \{\lambda^*\}$, then the gradient flow starting at $\lambda \in U$ converges to λ^*. By applying Theorem 5.3.2, it is easy to see that the algorithm is complete. □

5.4.3 Optimal structure in constant flow field with general convex Lagrangian

In general, if we only assume the Lagrangian $L = L(v)$ is a convex function and the dynamics is $\dot{x} = f(u+v)$ where f is invertible and 0 is in the range of f (there exists some y with $\|y\| < +\infty$ such that $f(y) = 0$), we can have similar optimal path within a constant

flow field and the result is stated in Theorem 5.3.1. In this section, we give the proof of this theorem.

Proof. Denoting $g(w)$ to be the inverse of f such that if $w = f(u+v)$ then $u+v = g(w)$, we will show that

$$\psi(x,t) = \begin{cases} tL\left(g\left(\frac{x-x_0}{t}\right) - u\right) & \left\|g\left(\frac{x-x_0}{t}\right) - u\right\| \le V \\ +\infty & \text{otherwise} \end{cases},$$

satisfies the HJB equation (5.15). First of all, the Hassian matrix $\mathcal{H}(v)$ is positive definite for all $\|v\| \le V$ since L is convex. Therefore, for any v_1, v_2 in the domain, there exists ξ such that

$$\nabla_v L(v_1) = \nabla_v L(v_2) + \mathcal{H}(\xi)(v_2 - v_1).$$

Further if $\nabla_v L(v_1) = \nabla_v L(v_2)$, it is true that $\mathcal{H}(\xi)(v_2 - v_1)$. Because of the positive definite property for H, we have $v_2 = v_1$, which implies that $\nabla_v L(v)$ is one-to-one.

Then we do the following calculation on the non-infinity part of ψ

$$\psi_t = L\left(g\left(\frac{x-x_0}{t}\right) - u\right) - \left[\nabla_v L\left(g\left(\frac{x-x_0}{t}\right) - u\right)\right]^T \nabla_w g\left(\frac{x-x_0}{t}\right)\frac{x-x_0}{t}, \tag{5.27}$$

$$\nabla\psi = \left[\nabla_w g\left(\frac{x-x_0}{t}\right)\right]^T \nabla_v L\left(g\left(\frac{x-x_0}{t}\right) - u\right). \tag{5.28}$$

Since L is convex, we further have the relaxed optimization

$$\max_v \{\nabla\psi^T(v+u) - L(v)\}$$

is a convex problem and get the condition for the optimal v^* to be

$$\begin{aligned}
&[\nabla_v f(u+v^*)]^T \nabla\psi = \nabla_v L(v^*) \\
\Longrightarrow &\nabla\psi = [\nabla_w g(f(u+v^*))]^T \nabla_v L(v^*).
\end{aligned}$$

Combining this with (5.28), we have

$$v^* = g\left(\frac{x-x_0}{t}\right) - u, \tag{5.29}$$

and $\|v^*\| \leq V$ holds. Thus, v^* is the maximizer of $H(x, \nabla\psi)$. Taking (5.27), (5.28) and (5.29), we have

$$\psi_t + \nabla\psi^T(v^*+u) - L(v^*) = 0,$$

implying that (5.15) holds. At last notice that $\lim_{t\to\infty} L\left(g\left(\frac{x-x_0}{t}\right) - u\right) = L(g(0)-u) < \infty$, since 0 is in the range of f. We have that $\lim_{t\to\infty} tL\left(g\left(\frac{x-x_0}{t}\right) - u\right) = \infty$. Thus, we have $t^* > 0$ such that given x,

$$t^* = \arg\min_{t\geq 0} \psi(x,t).$$

Thus, $v^* = g((x-x_0)/t^*) - u$ gives us a constant velocity motion. □

5.5 Simulation Results

In this section, we provide multiple simulations to validate the strength of the proposed method. First, the time-optimal and energy-optimal path planning examples with vehicle travel in simple canonical time flow field are presented. These examples serve as benchmark examples wherein, we compare the solution obtained by our algorithm to solutions derived analytically, or through numerical optimization, or other path planning methods. Then we present a path planning example of using the proposed method to plan the time-optimal and energy-optimal path in a realistic ocean surface flow field. This simulation is

intended to verify the performance of the proposed method in a highly complicated and strong real ocean flow field.

In this section, we consider the space to be $\mathbb{R}^2$ or $\mathbb{R}^3$. In $\mathbb{R}^2$, boundaries are line segments and the parametrization can be represented by the endpoints p_i^1, p_i^2 as $x_i = \lambda_i p_i^1 + (1-\lambda_i)p_i^2$ with $\lambda_i \in [0,1]$. Meanhwhile in $\mathbb{R}^3$, we take use of the plane equation $Ax + By + Cz = 0$ and by assuming $C \neq 0$, the parametrization is

$$\left(x, y, -\frac{Ax + By}{C}\right).$$

Moreover, if $C = 0$, we could choose the non-zeros one between A, B to have the parametrization.

5.5.1 Constant flow

In this case, we consider the vehicle travels in a constant flow field. The flow speed is $(1, 1)$ for all position in the domain. As shown in Fig. 5.2, though the flow field is uniform, the domain is divided into several regions of the same flow speed, in order to meet with the assumption that each of the regions is convex.

For the minimum time path planning, the proposed method generates one junction point at $(-0.0001, 9.5)$. By connecting the junction point with the starting and goal position, we can construct the time-optimal path, as is shown in the left figure in Fig. 5.2. The planned path is almost a straight line connecting the starting position to the goal position. Analytical solution of the time-optimal path can be derived from solution of the Zermelo's navigation problem [73], which is a straight line connecting the starting position to the goal position. In this case, the proposed method is able to derive the time-optimal path with numerical error at the scale of 10^{-4}, in a 20×20 domain. The scale of path planning error is related to the choice of parameter of the proposed algorithm, and can be further reduced.

In addition, the cost computed by the proposed method is identical to the cost computed

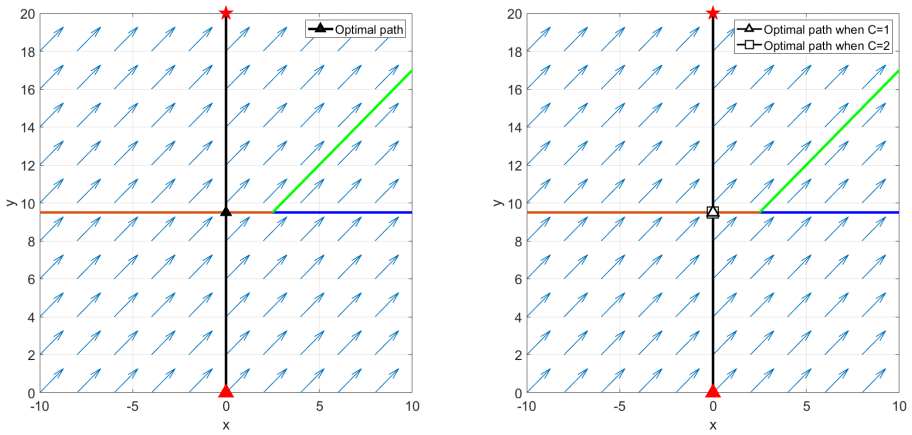


Figure 5.2: (Left) Time optimal path planned by the proposed method. (Right) Energy optimal path planned by the proposed method given different running cost C assigned. For both plots, the red triangle represents the starting position, while the red star denotes the goal position. Boundaries of the regions are denoted by colored solid lines. The flow speed at every position in the domain is marked by the blue arrows. The triangle and square markers denote the junction points position of each planned path, while the black line represents the optimal path.

analytically. Since the total vehicle speed $u + v$ aligns with the tangent direction of the planned path, the vehicle speed can be computed by the parallelogram law, which is $v = (-1, \sqrt{8})$. Therefore, the minimum travel time is 5.2241. The travel time computed by the proposed method is 5.2241, same as the analytical result. Thus, the proposed method finds the time-optimal path, and also accurately computes the optimal travel time in the constant flow case.

The right figure in Fig. 5.2 shows the energy-optimal path under different assigned running cost. For both $C = 1$ and $C = 2$ cases, the proposed method generates junction point at $(-0.0001, 9.5)$, same as the junction computed in the time-optimal path planning case. The straight planned path is also consistent with the theoretical analysis presented in [74].

In addition, the cost computed by the proposed method is also identical to the cost computed analytically. Since in the constant flow field, the vehicle travels in constant

speed, the problem can be reformulated as

$$\begin{aligned} \min_{T} \quad & (\|v\|^2 + C)T \\ s.t. \quad & (v+u)T = r_f - r_0, \end{aligned} \tag{5.30}$$

where r_f and r_0 denote the goal position and the starting position.

This optimization problem can also be solved analytically by taking the equality constraint into the optimization objective. Taking derivative of the optimization objective with respect to T. The cost computed analytically is 29.2820 when $C = 1$, while the proposed method also generates exactly the same result. The cost computed analytically is 40.0000 when $C = 2$, while the proposed method also generates cost of the same value.

5.5.2 Jet flow

In the second benchmark example, we apply the proposed path planning algorithm to let the vehicle optimally cross a jet flow. As shown in Fig. 5.3, we consider a flow field containing a uniform jet, from left to right, of constant speed $(2.9, 0)$, which is 0.1 less than the vehicle speed. Flow speed is assumed to be zero anywhere outside the jet flow.

For the minimum time path planning, the proposed method generates two junction points at $(-1.1180, 7.5)$ and $(1.4161, 10.5)$. The time-optimal path is shown in the left figure in Fig. 5.3. In order to analytically derive the time-optimal path, we adopt the same formulation as in [75], and formulate the time optimal path planning problem to be a nonlinear optimization problem. Parameters used for formulating the problem are described in Fig. 5.4. Thus, we have the nonlinear optimization problem formulated in (5.31).

$$\begin{aligned} \min \quad & T = \frac{y_1}{V\cos\theta_1} + \frac{d}{V\cos\alpha} + \frac{y_2}{V\cos\theta_2} \\ s.t. \quad & y_1\tan\theta_1 - d(\tan\alpha + \frac{u}{V}\sec\alpha) + y_2\tan\theta_2 = 0. \end{aligned} \tag{5.31}$$

The travel time is represented as the N-S displacement divided by vehicle's total speed in

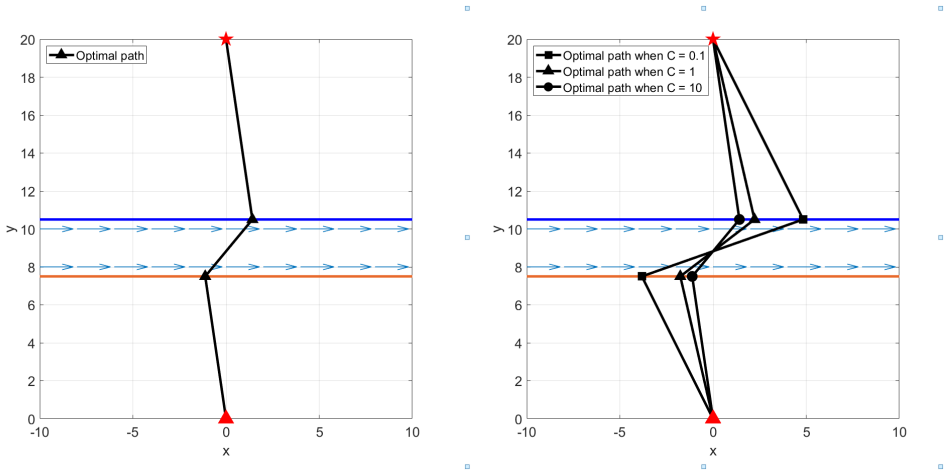


Figure 5.3: (Left) Time optimal path planned by the proposed method. (Right) Energy optimal path planned by the proposed method given different running cost C assigned. The red triangle and red star represent the starting and goal position respectively. Boundaries of the jet flow are denoted by colored solid lines. The jet flow speed is represented by the blue arrows. The optimal path is marked by black line, while the marker position denotes junction points computed by the proposed method.

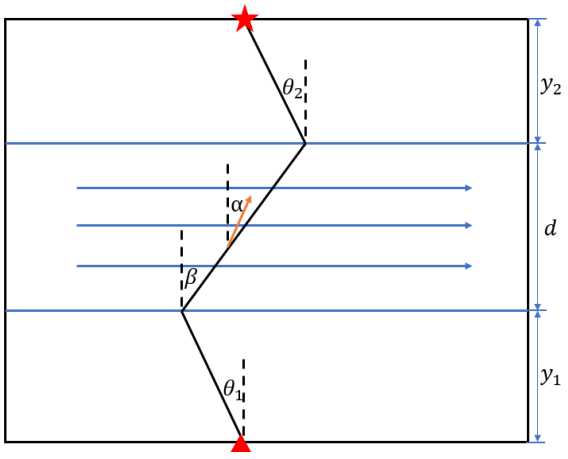


Figure 5.4: Demonstration of parameters used in formulating path planning into nonlinear optimization problem. $\theta_1, \theta_2, \beta$ denotes the angle of the three segments of planned path; α represents vehicle's steering angle when traveling inside the jet; d denotes the width of the jet flow, and y_1, y_2 describes the distance between boundaries of the jet and the starting and goal position.

N-S direction, and the travel time is minimized under the constraint that the horizontal displacement from starting to final position is zero. By solving the optimization numerically in MATLAB, we have the path planning results comparison in Table 5.1. Moreover, we have also applied LSM to the same time-optimal path planning problem, based on the LSM

toolbox described in [76], and presented parameters of the planned path in Table 5.1. From this table, if we consider the planned path derived by nonlinear optimization with MATLAB as ground truth, the proposed method achieves accuracy of scale 10^{-4} in a 20×20 domain. Compared with LSM, the proposed method can achieve higher accuracy in this canonical flow scenario.

For the minimum energy path planning, we generate the optimal path given three different choices of running cost. Since the cost function of minimizing energy is assigned as minimizing the integral of $\|v\|^2 + C$ over time, in order to derive representative cases of the selection of C, we choose $C = 0.1$, which is one magnitude smaller than vehicle speed, and vehicle speed overrides the running cost in the cost function; $C = 1$, which is at the same scale as vehicle speed; and $C = 10$, which is one magnitude larger than vehicle speed, and the running cost in the cost function dominants over the energy cost. The planned optimal paths are shown in the right figure of Fig. 5.3. When $C = 10$, the proposed method generates two junctions at $(-1.1185, 7.5000)$ and $(1.4156, 10.5)$. The two generated junctions are the same as the junctions of time-optimal path. This simulation result is consistent with the theoretical analysis presented in 6. In regions inside and outside the jet flow, according to Lemma 5.4.3, the vehicle will be traveling in its maximum speed, and the problem of energy optimal path planning is equivalent to the problem of time optimal path planning. Thus, energy-optimal path is the same as the time-optimal path, given $C = 10$.

When we select $C = 1$, the proposed method generates two junction points at $(-2.1772, 7.5000)$ and $(2.7578, 10.5000)$. We verify the accuracy of the proposed method on deriving the energy optimal path by comparing the planned path given $C = 1$ with the optimal path computed by MATLAB nonlinear optimization. The comparison result is shown in Table 5.2. If we consider the energy optimal path planned with MATLAB nonlinear optimization as ground truth, the proposed method also achieves accuracy of scale 10^{-4} in a 20×20 domain, in the given canonical flow field.

When $C = 0.1$ is chosen, the energy-optimal path has even more deviation from the

Table 5.1: Comparison of time-optimal planned path between using the proposed method, LSM, and MATLAB nonlinear optimization

	Proposed Method	LSM	Nonlinear Optimization with MATLAB
θ_1	$8.4785°$	$8.4113°$	$8.4782°$
β	$-40.1878°$	$-39.7923°$	$-40.1873°$
α	$-7.4134°$	$-8.1750°$	$-7.4143°$
θ_2	$8.4783°$	$8.2147°$	$8.4782°$

Table 5.2: Comparison of energy-optimal planned path between using the proposed method, and MATLAB nonlinear optimization given $C = 1$

	Proposed Method	Nonlinear Optimization with MATLAB
θ_1	$16.1877°$	$16.1873°$
β	$-58.7045°$	$-58.7045°$
α	$-9.9234°$	$-9.9234°$
θ_2	$16.1877°$	$16.1873°$

time-optimal path. Optimality verification of the planned path in this case is not included in the paper to avoid repetition.

Block flow

For the third benchmark example, we present a case where there are multiple optimal paths in a symmetrical flow field, in order to show that the proposed method is able to find all of the optimal paths in the domain with the help of intermittent diffusion technique. As shown in Fig. 5.5, the domain consists of five regions in total, with one region containing strong flow of speed $(0, -2)$, preventing the vehicle heading straight from the stating position to the goal position. There are two regions on the left and on the right of the strong flow region, of flow speed $(1.5, 0)$ and $(-1.5, 0)$, respectively. The two regions near the starting and goal position are of zero flow. Since the flow field is symmetric in the horizontal direction, there should be two symmetric time-optimal and energy-optimal paths. As shown in Fig. 5.5, the proposed method is able to derive two time-optimal paths and two energy-optimal paths.

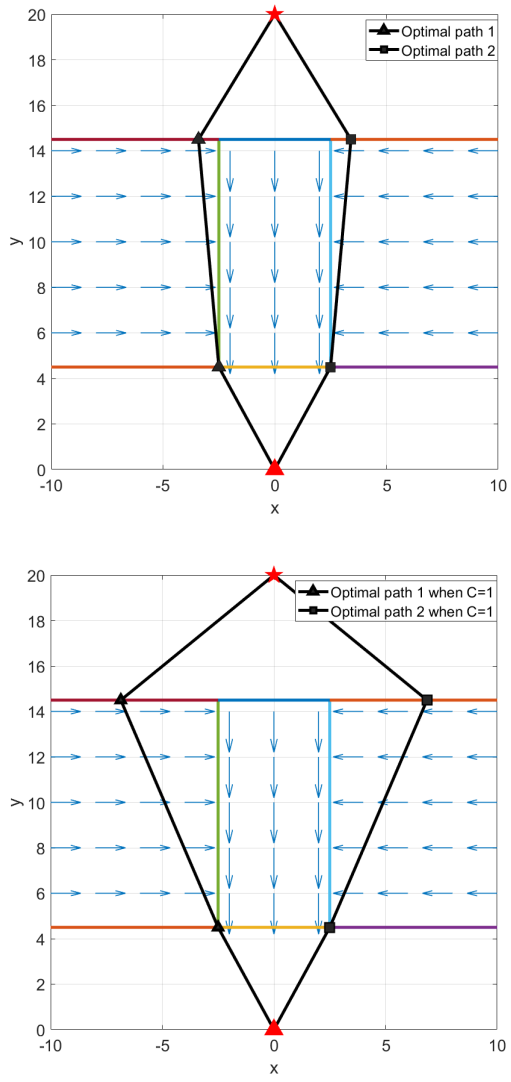


Figure 5.5: (Left) Time optimal paths planned by the proposed method. (Right) Energy optimal paths planned by the proposed method. For both time-optimal path planning and energy-optimal path planning, multiple optimal paths are generated. Boundaries of the jet flow are denoted by colored solid lines. The flow speed in the domain is represented by the blue arrows. The optimal path is marked by black line, while the marker position denotes junction points computed by the proposed method.

Jet flow in $3D$ *space*

For this benchmark example, we present path planning using the proposed method in a jet flow in $3D$ space. The domain consists of three regions, divided by two boundary surfaces, $z = 10$ and $z = 15$. In the region where $z \in (0, 10)$, the flow speed is $(0.5, 0, 0)$. There is

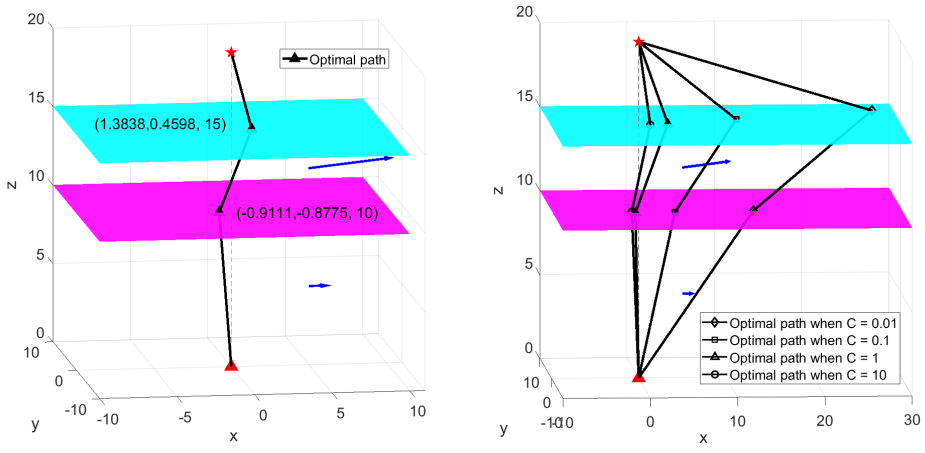


Figure 5.6: (Left) Time optimal paths planned by the proposed method. The two junction positions are marked on the plot (Right) Energy optimal paths planned by the proposed method. In both plots, boundaries of the jet flow are denoted by the colored surfaces. The flow speed in the domain is represented by the blue arrows. The optimal path is marked by black line, while the marker position denotes junction points computed by the proposed method.

strong jet flow in the region where $z \in (10, 15)$ with flow speed $(2, 1, 0)$. The flow speed is zero in the region where $z \in (15, 20)$. The starting position is assigned at the origin, while the goal position is assigned at $(0, 0, 20)$.

The left figure in Fig. 5.6 shows the time-optimal path planned by the proposed method. The time-optimal solution is compared with the time-optimal path planned by the LSM. For LSM, the domain size is set to be 80×80 grid cells, time step is 0.01. The comparison result is shown in Table 5.3. In this comparison, assuming the path segment $x_{i+1} - x_i$ travels from the boundary surface $f_{\alpha_i\beta_i}$ to reach the boundary surface $f_{\alpha_{i+1}\beta_{i+1}}$, we define θ_i as the angle between path segment $x_{i+1} - x_i$ and the boundary surface $f_{\alpha_i\beta_i}$. γ_i is defined as the angle between the projection of $x_{i+1} - x_i$ on the boundary surface $f_{\alpha_i\beta_i}$ and the x-axis of $f_{\alpha_i\beta_i}$, $\theta_i \in (0, 90°], \gamma_i \in (-180°, 180°]$. From the table, θ_i and γ_i computed from the proposed method and the LSM are similar, with approximately $1°$ difference. Travel time of the optimal path planned by the proposed method and LSM are also approximately the same. The travel time of optimal path planned by LSM is 0.073 larger than the travel time of the optimal solution from the proposed method.

Table 5.3: Comparison between using the proposed method, and LSM for time-optimal path planning

	Proposed Method	LSM
θ_1	82.7924	83.5659
θ_2	62.0255	63.3118
θ_3	73.7397	73.8027
γ_1	-136.0775	-135.6592
γ_2	30.2293	30.2407
γ_3	-161.6199	-161.2246
Travel time	6.9096	6.9826
Computation time	0.57 secs	10125.2 secs

Though the optimal solution generated by the two methods are almost the same, the computation time differs significantly. In this flow field setting, LSM takes considerably higher computation cost compared with the proposed method, as shown in Table 5.3. The proposed method computes the optimal path by searching for junctions that minimize the cost function only on the boundaries of flow regions. LSM solves for the optimal path by propagating the reachability front, and the computation of reachability front is done by embedding it as the zero-level set of a higher dimensional function [77]. Therefore, in $3D$ case, the computational labor $\mathcal{O}(n^3)$ is required per time step, where n is the number of grid points in each of the spatial direction [77]. This explains the significant difference in computational cost of the two methods in this flow field setting.

Note that for this benchmark example, the MATLAB nonlinear optimization method is not applied. The optimal solution in this benchmark example is close to a singular point of the optimization problem, introducing difficulty in solving for the optimal solution.

The energy-optimal path is shown in the right figure in Figure 5.6. The energy-optimal path when $C = 10$ is exactly the same as the time-optimal path. As the assigned C decreases, the energy cost is attached relatively more weight in the cost function. Therefore, the vehicle tends to save more energy to go with the flow in the bottom region and the jet flow region. Therefore, the energy-optimal path deviates from the time-optimal path as C decreases.

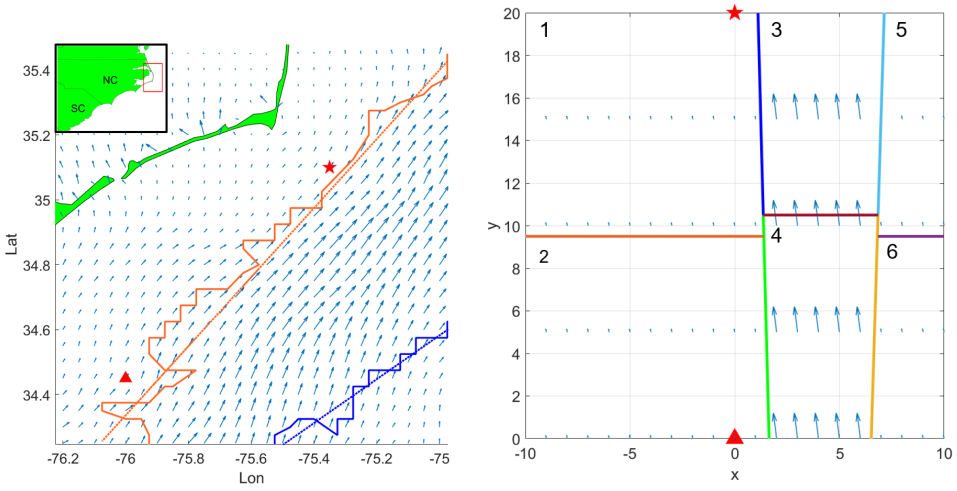


Figure 5.7: (Left) Surface ocean flow field on May 27, 2017, 00:00 UTC at Cape Hatteras, NC. The orange and blue curved lines indicates the boundaries of the separated regions computed from K-Means method. The orange and blue straight lines indicates the smoothed boundaries of the regions derived by fitting the curved boundaries into straight lines. Red triangle and red star indicate the starting and goal position. (Right) Rotated and rescaled deployment domain divided into regions of uniform flow. Index of regions are marked at the top-left corner of each region.

5.5.3 Surface ocean flow

In this section we present path planning simulation of glider traveling in real ocean surface flow field near Cape Hatteras, North Carolina. The input flow map for path planning is given by a 1-km horizontal resolution version of the Navy Coastal Ocean Model (NCOM) [78] made available by J. Book and J. Osborne (Naval Research Laboratory, Stennis Space Center). We will first describe using K-means clustering analysis method for flow field partitioning, then the results of applying the proposed path planning method on the partitioned flow field will be presented.

Flow field partition algorithm

Given a predefined origin of the domain of vehicle deployment, the flow field data input on the uniform Longitude and Latitude grid is converted using Mercator projection to flow data on non-uniform Cartesian coordinate, where the x-axis denotes the Eastward distance

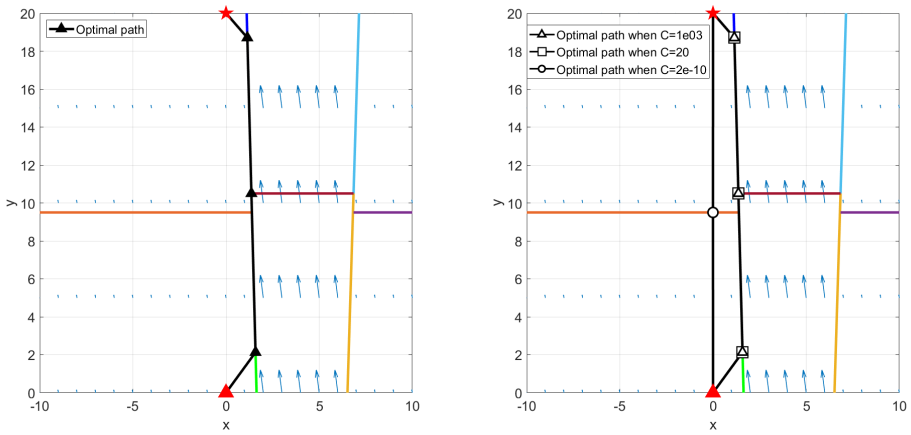


Figure 5.8: (Left) Time-optimal path. (Right) Energy-optimal path given different running cost. The optimal path is marked by black line, while the marker position denotes junction points computed by the proposed method.

to the origin, and the y-axis denotes the Northward distance to the origin. Let $x \in \mathbb{R}^2$ denote the position of all flow data in Cartesian coordinate, while $u \in \mathbb{R}^2$ represent the Eastward and Northward flow speed. Let $y \in \mathbb{R}^4$,

$$y = \left[\frac{\|x(1)\|}{\max(\|x(1)\|)}, \frac{\|x(2)\|}{\max(\|x(2)\|)}, \frac{u(1)}{\max(\|u(1) + iu(2)\|)}, \frac{u(2)}{\max(\|u(1) + iu(2)\|)} \right],$$

denote the vector observation on each grid cell in the domain normalized to the scale of $[0, 1]$. Then the domain can be divided into regions, $S = \{S_1, S_2, ..., S_k\}$ by finding the minimum of the following function

$$\arg\min_{S} \sum_{i=1}^{k} \sum_{y \in S_i} \|y - \mu_i\|^2,$$

where μ_i denotes the centroid of points in S_i. The above optimization problem can be solved by the K-means method. It iteratives between two steps. The first step assigns each data point in the domain to its nearest centroid, while the second step recomputes the centroid position of each cluster by averaging the data points assigned to the centroid's cluster. The algorithm terminates when the assignment of data points no longer changes.

Then $S = \{S_1, S_2, \ldots, S_k\}$ is the derived flow field partition.

Next we will introduce the derivation of boundary points, given the partitioned flow regions. Let $x_{(i,j)}$ denote the position of one data point in the cartesian coordinate. Define the set of neighboring points of (i, j) as $N(i, j) = \{(i \pm 1, j), (i, j \pm 1)\}$. If $(i, j) \in S_\alpha$, while there exists data point $(m, n) \in N(i, j)$ that belongs to region $S_\beta, \beta \neq \alpha$, then $\frac{1}{2}(x_{(i,j)} + x_{(m,n)})$ is included in the boundary set between region α and region β. By iterating this process on all grid cells in the domain, all boundaries between regions can be detected.

Since the crooked flow region boundaries may introduce excessive difficulty to applying the proposed path planning method, we introduce another step of smoothing the boundaries of partitioned flow regions. For boundary between region α and region β, denoted as $f_{\alpha\beta}$, we apply the following Least Mean Square method to fit the boundary curves into straight lines.

$$\arg\min_{a,c} \sum_{b_p \in f_{\alpha\beta}} dist(ax + y + c = 0, b_p)$$

where the distance function $dist(ax + y + c = 0, b)$ is defined as the Euclidean distance between the line defined by function $ax + y + c = 0$ and the point b_p.

Flow field partition and path planning results

Boundaries of the divided regions are shown in the left figure of Fig. 5.7. As shown in this figure, from K-Means method, the flow field is divided into three regions, a strong jet flow region, and two regions outside the strong flow. The crooked flow boundaries are smoothed into straight lines using the Least Mean Square method. By rescaling, and rotating the flow field according to the starting and goal position, we transformed the domain into the field shown in the right figure of Fig. (5.7). Note that in this figure, several new regions are added in order to simplify the computation of the proposed method. Accordingly, the flow speed and vehicle speed are also rescaled. Glider's horizontal forward speed is $0.27 - 0.3ms^{-1}$ on averaged. Thus, the rescaled vehicle speed is $6.51e{-}5$. Similarly,

after rescaling and rotating, the strong jet flow region is of speed $(-5.91e{-}5, 6.05e{-}4)$, which is about ten times faster than the vehicle speed. Region 1 and Region 2 are of flow speed $(-4.26e{-}6, 7.24e{-}5)$, approximately at the same magnitude as the vehicle speed. Region 5 and Region 6 are of flow speed $(-3.31e{-}6, 7.92e{-}5)$, also at the same scale of the rescaled vehicle speed.

The time-optimal and energy-optimal paths are shown in Fig. 5.8. For the time-optimal path, the vehicle makes a detour and takes advantage of the strong vertical ocean flow to travel to the goal position. Energy optimal path planning generates similar results when $C = 1e03$ and $C = 20$. In these cases, the running cost is much larger than the vehicle speed. Thus the energy-optimal planned path is identical to the time-optimal path. When $C = 2e - 10$, the running cost is much less than vehicle speed. In this case, instead of making use of the strong jet flow, the proposed method generates planned path that go straight towards the goal position.

5.6 Conclusion

In this paper, we propose a new method using Method of Evolving Junctions to solve the AUV path planning problem in an arbitrary flow field with the dynamics being $\dot{x} = u + v$ where u is the flow field and v is the vehicle velocity. Taking advantage of the explicit solution in constant flow field being straight line motion, we partition the flow field into piece-wise constant vector field and transform the optimal control problem into a finite dimensional optimization, using intermittent diffusion method to get the global minimizer. In this way, we can get rid of the system error induced by discretizing the continuous space. Also, our method can be trivially extended to high dimensional general vehicle path planning problems in the same time complexity without making further assumption.

CHAPTER 6

K-MEANS ON GRAPHS VIA OPTIMAL TRANSPORT

6.1 Introduction

K-means is an important method in data analysis, seeking a set of centroids, which is used to partition a given dataset into finite number of classes. Typically Euclidean distance is taken for centroids selection [79, 80, 81] and Lloyd's algorithm is an efficient method to handle this case, which gives meaningful local optimal, although in finding the global optimal solution of the K-means problem is NP-hard [82, 83]. K-means++ method [84] gives an efficient way to create initializations and several other strategies have been proposed to speed up the computations [85, 86]. In [87], a backward Euler method is proposed to solve K-means problem. If outliers exist, local search related method is embedded in the algorithm [88]. And [89] improves K-means algorithm if the dataset is large.

Meanwhile, optimal transport theory studies how to transport one density to another on the probability space and defines the Wasserstein distance [90]. It has been used for clustering problem involves distributions [91, 92, 93], especially for interval data and histograms [94, 95, 96]. It is worth mentioning that the Wasserstein distance admits an explicit formula for one-dimensional histogram [97].

In practice, the dataset often lives on a discrete graph instead of continuous space and optimal transport provides powerful tools incorporating the graph structure [98, 99, 100, 101]. Likewise in the text classification problem [102, 103, 104], one treats each document as a sample generated independently from multinomial distributions and needs to cluster bunches of documents into several classes with distribution centroids calculated. Thus, solving the clustering problem on a graph is in demand.

In this chapter, motivated by [105, 106], we define the discrete Wasserstein-$(1, p)$ dis-

tance on a simple finite graph $G = (V, E)$. Here $(1, p)$ represents the ground metric on the graph is l_p, which is homogeneous degree one. With this distance function on a graph, we propose the K-means problem on the discrete probability simplex. It aims to cluster different distributions supported on G to a fixed number of classes. To efficiently solve this minimization, we apply the gradient flow induced by discrete Wasserstein-2 metric [53, 107]. Our framework follows the Lloyd's algorithm. It contains two major steps: assign each data point to a class with given centroids and re-calculate centroids locations within every class. The first step only involves the calculation of Wasserstein-$(1, p)$ distance. For the second part, we formulate the problem in an optimization and compute the problem efficiently using the gradient flow induced by Wasserstein-2 metric on graphs.

This chapter is organized as below: In Section 6.2, we propose the Wasserstein K-means problem on the probability space. In Section 6.3, we introduce the definition of discrete Wasserstein-$(1, p)$ distance with $p \in [1, \infty]$ as well as the gradient flow of the distance function induced by Wasserstein-2 metric. Then the algorithm is given and corresponding convergence analysis is stated in Section 6.3.4. Following this, experiments are conducted in Section 6.4, within which, the results of Wasserstein-$(1, p)$ calculations are shown with different orientations and graph structures in Section 6.4.1. In Section 6.4.2, 6.4.3 and 6.4.4, we illustrate on how the Wasserstein-2 gradient flow works on distinct graphs. At last, we experiment on our algorithm with two K-means problems in Section 6.4.5 and 6.4.6 and finish this chapter with a brief conclusion.

6.2 Problem Statement

Consider a connected simple graph $G = (V, E)$ with V the vertex set and E the edge set. The goal here is to separate a set of discrete distributions $D = \{\rho^\beta\}_{\beta=1}^n \subset \mathcal{P}(G)$ into K clusters, where $n < \infty$ and $\mathcal{P}(G) = \{(\rho_i)_{i\in V} : \sum_{i=1}^{|V|} \rho_i = 1, \rho_i \geq 0\}$ is the probability simplex supported on the nodes of G.

This problem can be formulated into a K-means problem. Given a distance function

$d(\cdot,\cdot)\colon \mathcal{P}(G)\times\mathcal{P}(G)\to\mathbb{R}$, consider

$$\arg\min_{S}\sum_{\alpha=1}^{K}\sum_{\rho\in S^{\alpha}} d(\rho, c^{\alpha}), \tag{6.1}$$

where c^{α} is the centroid of cluster S^{α}, $S=(S^1,\cdots,S^K)$ and

$$S^{\alpha}=\{\rho\in D : \rho \text{ is in the } \alpha \text{ cluster}\}.$$

A classical approach applies the Euclidean distance as the distance function, and solves the K-means problem over two steps, known as expectation (E)–maximization (M) algorithm. The first step (E step) assigns each $\rho\in D$ to the cluster α by solving

$$\alpha=\arg\min_{\gamma\in\{1,\cdots,K\}} d(\rho, c^{\gamma}). \tag{6.2}$$

And the second step (M step) uses the updated cluster information to relocate the centroids for each cluster:

$$c^{\alpha}=\arg\min_{x\in\mathcal{P}(G)}\sum_{\rho\in S^{\alpha}} d(x,\rho). \tag{6.3}$$

In this chapter we specify the distance function d as the discrete Wasserstein-$(1,p)$ ($W_{1,p}$) distance

$$W_{1,p}(\rho^0,\rho^1)=\inf_{u}\left\{\|u\|_{1,p} : div(u)+\rho^1-\rho^0=0\right\}. \tag{6.4}$$

Here $p\in[1,+\infty]$, $\nabla\in\mathbb{R}^{|E|\times|V|}$ is the discrete gradient operator, which has many different ways to be defined. We give one used in this chapter in Section 6.3. $div=-\nabla^T$ is the discrete divergence operator, u is the flux function defined on E and the infimum is taken among all discrete flux with linear constraints involving two discrete densities ρ^0,ρ^1. Especially, we are interested in the case when $p=2$, which has the ground metric to be standard Euclidean metric since it is the most commonly used distance for K-means

problem.

In the algorithm, the E step calculates the $W_{1,p}$ distances from the current centroids to the data points and assign each data point to the cluster with the shortest distance. In this procedure, the minimization problem (6.4) is involved to compute $W_{1,p}$. Meanwhile, the M step re-assign the centroids for each cluster by solving another minimization (6.3). Notice that (6.3) does not admit an explicit solution, we apply the gradient flow of this objective function induced by Wasserstein-2 (W_2) metric on graphs to handle this problem.

6.3 Algorithm

In this section, we define Wasserstein-$(1,p)$ distances on graphs, then compute the differential of the Wasserstein-$(1,p)$ distance, and at last introduce the discrete W_2 gradient descent method. Using them, we provide the algorithms with convergence results.

6.3.1 Discrete $W_{1,p}$ Distance

Given a continuous space $\mathcal{M}$, the Wasserstein $(1,p)$ distance on $\mathcal{P}(\mathcal{M})$ is

$$W_{1,p}(\nu_0,\nu_1) = \inf_{\pi\in\Pi(\nu_0,\nu_1)} \iint_{\mathcal{M}\times\mathcal{M}} \|x-y\|_p d\pi(x,y)$$

where $\|\cdot\|$ is the l_p-norm, i.e.

$$\|x-y\|_p = \Big(\sum_{i=1}^{n} |x_i-y_i|^p\Big)^{\frac{1}{p}},$$

and

$$\Pi(\nu_0,\nu_1) = \{\pi : \pi(A\times\mathcal{M}) = \nu_0(A), \pi(\mathcal{M}\times B) = \nu_1(B)\},$$

for all measurable set $A, B \subset \mathcal{M}$. Assuming that ν_0 and ν_1 are absolutely continuous with respect to the Lebesgue measure vol, we introduce the density function for ν_0 and ν_1, denoting them as $d\nu_0 = \rho^0 dvol$, $d\nu_1 = \rho^1 dvol$. We can rewrite the distance function by the

dual formulation [108, 109, 110, 90]:

$$W_{1,p}(\rho^0,\rho^1) = \inf_{v,\rho}\Big\{\int_0^1\int_{\mathcal{M}} \|v(t,x)\|_p \rho(x,t)dxdt : \\ \partial_t\rho(t,x) + div(\rho(t,x)v(t,x)) = 0, \\ \rho(\cdot,0)=\rho^0, \rho(\cdot,1)=\rho^1\Big\},$$

where the infimum is taken among all vector field $v(t,x)$ and density function $\rho(t,x)$ such that the continuity equation with zero flux boundary condition holds. Furthermore, denote $u(x) = \int_0^1 \rho(t,x)v(t,x)dt$ as the flux function, following the Jensey's inequality, the above optimal control formulation in density space is equivalent to the minimization problem [105]:

$$W_{1,p}(\rho^0,\rho^1) = \inf_{u}\Big\{\int_{\mathcal{M}} \|u(x)\|_p dx : \\ \text{div}(u(x)) + \rho^1(x) - \rho^0(x) = 0, \\ u(x)\cdot \mathbf{n}(x) = 0 \text{ for all } x\in\partial\mathcal{M}\Big\}, \tag{6.5}$$

where $\mathbf{n}$ is normal to $\partial\mathcal{M}$ and the infimum is over all flux function $u(x)$.

Now we consider the case $\mathcal{M} = G$. We define the Wasserstein metric on a graph through formulation (6.5). We assign an orientation on G, which later brings us convenience to use the Bregman iteration and explicit update formulation for gradient flow. On each edge $(i,j)\in E$, we specify a direction either from i to j or the opposite and we denote $(i,j)\in E$ to imply the direction as from i to j, otherwise we use (j,i) to represent this edge. With this, $\nabla = (\nabla_{ij,k})_{(i,j)\in E,k\in V} \in \mathbb{R}^{|E|\times|V|}$ is defined by

$$\nabla_{ij,k} = \begin{cases} 1 & k = i \\ -1 & k = j \\ 0 & k\neq i,j \end{cases}$$

With the orientation, we denote

$$u_i = \left(u_{(i,j)}\right)_{j \in \vec{N}(i)}, \tag{6.6}$$

where $\vec{N}(i) = \{k \in V : (i,k) \in E\}$. In this way, we introduce the following l_p norm on the graph

$$\int_{\mathcal{M}} \|u\|_p dx = \|u\|_{1,p} = \sum_{i \in V} \|u_i\|_p$$

for $p \in [1, +\infty]$. It is straightfoward to show that the proposed method is a consistent discreteization for the problem (6.5). The minimization problem is now in the form of (6.4). To solve (6.4), we make use of the Lagrangian multiplier

$$L(u, \Phi) = \|u\|_{1,p} + \Phi^T (div(u) + \rho^1 - \rho^0)$$

and derive the saddle point problem

$$W_{1,p}(\rho^0, \rho^1) = \min_u \max_\Phi L(u, \Phi). \tag{6.7}$$

Saddle problem (6.7) can be solved by the first-order primal-dual method [105, 111, 112]:

$$u^{k+1} = \arg\min_u \left\{ \|u\|_{1,p} + (\Phi^k)^T div(u) + \frac{1}{2\mu} \|u - u^k\|_2^2 \right\}, \tag{6.8}$$

$$\phi^{k+1} = \arg\max_\Phi \left\{ \Phi^T \left(div(2u^{k+1} - u^k) + \rho^1 - \rho^0 \right) - \frac{1}{2\tau} \|\Phi - \Phi^k\|_2^2 \right\}, \tag{6.9}$$

where $\|\cdot\|_2$ is the standard Euclidean metric with μ and τ being two predefined parameters. Here formula (6.8) represents the discrete gradient descent direction, while formula (6.9) forms the discrete gradient ascent direction.

It turns out we can get an explicit formula for the update of u^{k+1} and Φ^{k+1}. For u from

(6.8),

$$\begin{aligned}
&\arg\min_u \left\{ \|u\|_{1,p} + (\Phi^k)^T div(u) + \frac{1}{2\mu}\|u-u^k\|_2^2 \right\} \\
=&\arg\min_u \left\{ \sum_{i\in V} \|u_i\|_p - (\nabla\Phi^k)u + \frac{1}{2\mu}\|u-u^k\|_2^2 \right\} \\
=&\arg\min_u \left\{ \sum_{i\in V} \|u_i\|_p - \sum_{(i,j)\in E} (\nabla\Phi^k)_{(i,j)} u_{(i,j)} + \frac{1}{2\mu}\sum_{(i,j)\in E} (u_{(i,j)} - u^k_{(i,j)})^2 \right\} \\
=&\arg\min_u \left\{ \sum_{i\in V} \left(\|u_i\|_p - (\nabla\Phi^k)_i^T u_i + \frac{1}{2\mu}(u_i - u_i^k)^2 \right) \right\},
\end{aligned}$$

with $u_i, (\nabla\Phi^k)_i, u_i^k$ in the sense of (6.6).

Thus, updating u is the same as updating u_i. With the results of the proximal operator, we have

$$u_i^{k+1} = shrink_p(u_i^k + \mu(\nabla\Phi^k)_i, \mu) \text{ for } i \in V.$$

where $p \in (1, +\infty]$ and with q satisfying $p^{-1} + q^{-1} = 1$, we have

$$shrink_p(v,\mu) = \max(\|v\|_q - \mu, 0)\frac{v}{\|v\|_q}.$$

If $p = 1$, we can further simplify (6.8) to

$$\arg\min_u \left\{ \sum_{(i,j)\in E} \left(\|u_{(i,j)}\|_p - (\nabla\Phi^k)_{(i,j)} u_{(i,j)} + \frac{1}{2\mu}\|u_{(i,j)} - u^k_{(i,j)}\|_2^2 \right) \right\},$$

which results in

$$u_{(i,j)}^{k+1} = shrink_1(u_{(i,j)}^k + \mu(\nabla\Phi^k)_{(i,j)}, \mu) \text{ for } (i,j) \in E$$

with

$$shrink_1(v,\mu) = \max(|v| - \mu, 0)\frac{v}{|v|}.$$

As for Φ, from (6.9), we directly have

$$\Phi^{k+1} = \Phi^k + \tau \left(div(2u^{k+1} - u^k) + \rho^1 - \rho^0 \right).$$

The algorithm to compute the $W_{1,p}$ distance is given in Algorithm 9.

6.3.2 Differential of $W_{1,p}$

With the definition of $W_{1,p}$, we can have the dual formulation. And the dual form gives the differential of $W_{1,p}$, which will be used in the computation of W_2 gradient flow later on.

Theorem 6.3.1. *Denoting (u^*, Φ^*) as the saddle point of L with ρ^0, ρ^1 as prior, we have*

$$\frac{\partial}{\partial \rho^0} W_{1,p}(\rho^0, \rho^1) = -\Phi^* \tag{6.10}$$

Proof. We first calculate the variational formula for L around the saddle point given arbitrary perturbation u^0 and $p^{-1} + q^{-1} = 1$:

$$\begin{aligned}
\left(\nabla_u L(u^*, \Phi^*)\right)^T u^0 &= \lim_{\delta \to 0} \frac{1}{\delta} \left(L(u^* + \delta u^0, \Phi^*) - L(u^*, \Phi^*) \right) \\
&= \lim_{\delta \to 0} \frac{1}{\delta} \left(\|u^* + \delta u^0\|_{1,p} - \|u^*\|_{1,p} + (\Phi^*)^T div(\delta u^0) \right) \\
&= \lim_{\delta \to 0} \frac{1}{\delta} \left(\|u^* + \delta u^0\|_{1,p} - \|u^*\|_{1,p} \right) - (\nabla \Phi^*)^T u^0 \\
&= \sum_{i \in V} \lim_{\delta \to 0} \frac{1}{\delta} \left(\|u_i^* + \delta u_i^0\|_p - \|u_i^*\|_p \right) - (\nabla \Phi^*)^T u^0 \\
&= \sum_{i \in V} \left(\partial_{j \in \vec{N}(i)} \|u_i^*\|_p \right)^T u_i^0 - (\nabla \Phi^*)^T u^0 \\
&= \sum_{i \in V} \frac{\sum_{j \in \vec{N}(i)} |u^*_{(i,j)}|^{p-1} u^0_{(i,j)} sign(u^*_{(i,j)})}{\|u_i^*\|_p^{p/q}} - (\nabla \Phi^*)^T u^0 \\
&= \sum_{(i,j) \in E} \left(\frac{|u^*_{(i,j)}|^{p-1} sign(u^*_{(i,j)})}{\|u_i^*\|_p^{p/q}} - (\nabla \Phi^*)_{(i,j)} \right) u^0_{(i,j)},
\end{aligned}$$

where

$$sign(x) = \begin{cases} 1 & x \geq 0 \\ -1 & x < 0 \end{cases}.$$

Since u^0 is arbitrary and (u^*, Φ^*) is saddle point, $\left(\nabla_u L(u^*, \Phi^*)\right)^T u^0 = 0$ and further we have for all $(i,j) \in E$

$$(\nabla \Phi^*)_{(i,j)} = \frac{|u^*_{(i,j)}|^{p-1} sign(u^*_{(i,j)})}{\|u_i^*\|_p^{p/q}} = \frac{|u^*_{(i,j)}|^{p/q} sign(u^*_{(i,j)})}{\|u_i^*\|_p^{p/q}}.$$

Therefore, we can have the dual form of (6.7) as

$$\begin{aligned}
W_{1,p}(\rho^0, \rho^1) &= L(u^*, \Phi^*) \\
&= \|u^*\|_{1,p} + (\Phi^*)^T \left(div(u^*) + \rho^1 - \rho^0\right) \\
&= \|u^*\|_{1,p} - (\nabla \Phi^*)^T u^* + (\Phi^*)^T (\rho^1 - \rho^0) \\
&= \|u^*\|_{1,p} - \sum_{(i,j)\in E} (\nabla \Phi^*)_{(i,j)} u^*_{(i,j)} + (\Phi^*)^T (\rho^1 - \rho^0) \\
&= \|u^*\|_{1,p} - \sum_{i \in V} \frac{\sum_{j \in \vec{N}(i)} |u^*_{(i,j)}|^p}{\|u_i^*\|_p^{p/q}} + (\Phi^*)^T (\rho^1 - \rho^0) \\
&= \|u^*\|_{1,p} - \sum_{i \in V} \frac{\|u_i^*\|_p^p}{\|u_i^*\|_p^{p/q}} + (\Phi^*)^T (\rho^1 - \rho^0) \\
&= \|u^*\|_{1,p} - \sum_{i \in V} \|u_i^*\|_p + (\Phi^*)^T (\rho^1 - \rho^0) \\
&= (\Phi^*)^T (\rho^1 - \rho^0),
\end{aligned}$$

which, accompanied with the observation that $\|(\nabla \Phi^*)_i\|_q = 1$ for all $i \in V$, leads to the fact that

$$W_{1,p}(\rho^0, \rho^1) = \inf_{\Phi} \left\{ \Phi^T (\rho^1 - \rho^0) : \|(\nabla \Phi)_i\|_q \leq 1, \forall i \in V \right\}. \tag{6.11}$$

Thus, combined with the envelope theorem [113], (6.11) implies (6.10). □

With this result, we can now move on to solve (6.3) using Wasserstein-2 gradient flow.

6.3.3 W_2 Gradient Flow

Having the $W_{1,p}$ and corresponding differential, we finally give the W_2 gradient flow scheme on graph. It is known that given a functional $\mathcal{F}(\rho)$ on the probability manifold of a continuous space, the W_2 gradient flow is [90]:

$$\rho_t = -grad\mathcal{F} = div\left(\rho\nabla\frac{\delta\mathcal{F}}{\delta\rho}\right) \tag{6.12}$$

where $\frac{\delta}{\delta\rho}\mathcal{F}$ is the gradient function of $\mathcal{F}$ with respect to the standard L^2 Euclidean structure.

Based on (6.12), we give the W_2 gradient flow on a graph G as

$$\rho_t(x,t) = -L(\rho)\nabla_\rho\mathcal{F}(\rho) \tag{6.13}$$

where

$$\nabla_\rho\mathcal{F}(\rho) = \left(F_1, \cdots, F_{|V|}\right),$$

with $F_i = \frac{\partial}{\partial\rho_i}\mathcal{F}$. We denote $N(i) = \{j : (i,j) \in E \text{ or } (j,i) \in E\}$ the neighborhood of i, then $L(\rho)$ is a projection matrix with each entry defined as

$$L(\rho)_{ij} = \begin{cases} \omega_{ij}\rho_i & F_i > F_j, j \in N(i) \\ \omega_{ij}\rho_j & F_i < F_j, j \in N(i) \\ \omega_{ij}\frac{\rho_i+\rho_j}{2} & F_i(\rho) = F_j(\rho), j \in N(i) \\ 0 & j \notin N(i) \end{cases}$$

providing that $\omega_{ij} = \omega_{ji} \geq 0$ as a set of weights on the edges.

Now to solve (6.3), given a specific cluster S_c we have our objective functional to be

$$\mathcal{F}_p^\alpha(\rho) = \sum_{\hat{\rho}\in S^\alpha} W_{1,p}(\rho,\hat{\rho}).$$

From Section 6.3.2, we conclude that

$$\frac{\delta}{\delta\rho}\mathcal{F}_p^\alpha(\rho) = \sum_{\hat{\rho}\in S^\alpha} \Phi^*(\rho,\hat{\rho}),$$

where $\Phi^*(\rho,\hat{\rho})$ is the saddle point of $W_{1,p}(\rho,\hat{\rho})$ in (6.7). We update the centroid of S^α with the stationary solution of

$$\begin{aligned}\dot{\rho}_i = &\sum_{j\in N(i)} \omega_{ij}\rho_j \left(\sum_{\hat{\rho}\in S^\alpha} \left(\Phi_j^*(\rho,\hat{\rho}) - \Phi_i^*(\rho,\hat{\rho})\right)\right)_+ \\ &- \sum_{j\in N(i)} \omega_{ij}\rho_i \left(\sum_{\hat{\rho}\in S^\alpha} \left(\Phi_i^*(\rho,\hat{\rho}) - \Phi_j^*(\rho,\hat{\rho})\right)\right)_+ .\end{aligned}$$

where $i \in V$. We use the forward Euler method to discretize in time. The fully discrete version of the gradient flow is

$$\begin{aligned}\rho_i^+ = \rho_i + &\Delta t \sum_{j\in N(i)} \omega_{ij}\rho_j \left(\sum_{\hat{\rho}\in S^\alpha} \left(\Phi_j^*(\rho,\hat{\rho}) - \Phi_i^*(\rho,\hat{\rho})\right)\right)_+ \\ &- \Delta t \sum_{j\in N(i)} \omega_{ij}\rho_i \left(\sum_{\hat{\rho}\in S^\alpha} \left(\Phi_i^*(\rho,\hat{\rho}) - \Phi_j^*(\rho,\hat{\rho})\right)\right)_+ .\end{aligned} \tag{6.14}$$

Remark 7. *Similar to the projection $L(\rho)$ in* (6.13)*, we can use the standard L^2 projection to derive the L^2 gradient flow:*

$$\dot{\rho}_i = -\left(F_i(\rho) - \sum_{j\in V} F_j(\rho)\right), \text{ for } i \in V, \tag{6.15}$$

and also the projection induced by Fish-Rao metric

$$\dot{\rho}_i = -\rho_i\left(F_i - \sum_{j\in V} F_j\rho_j\right), \text{ for } i \in V. \tag{6.16}$$

The convergence comparisons between these gradient flows will be provided in section 6.4.

Algorithm 8: K-means on graphs

Data: the graph $G = (V, E)$; density set $D = \{\rho^\beta\}_{\beta=1}^n$, cluster number K

Randomly choose K initial centroids $\{c^\alpha\}_{\alpha=1}^K \subset \mathcal{P}(G)$;

while *not converge* **do**

  Class Assignment: Assign each ρ^β to the cluster α (let $\rho^\beta \in S^\alpha$) with Algorithm 9 to find (6.2);

  Centroids Calculation: For each class S^α calculate the new centroid by solving (6.3) with Algorithm 10;

end

return *the centroid set* $\{c^\alpha\}_{\alpha=1}^K$;

Remark 8. *The Wasserstein K-means problem originally is defined on an undirected graph. The gradient flow induced by discrete Wasserstein-2 metric is as well orientation free. For discrete $W_{1,p}$ distance, there can be many definitions. However, in this chapter, we define the distance on some specific orientation just to make the calculation possible.*

6.3.4 Algorithm and Convergence Analysis

In this section, we give three main algorithms: $W_{1,p}$ distance calculation as Algorithm 9, the gradient flow of (6.3) as Algorithm 10 and the algorithm to solve K-means problem on $\mathcal{P}(G)$ as Algorithm 8.

The convergence of Algorithm 9 is guaranteed by the following theorem in [105]:

Theorem 6.3.2. *Assume $\tau\mu < 1/\lambda_{\max}(\nabla^T\nabla)$, where $\lambda_{\max}(\nabla^T\nabla)$ denotes the largest eigenvalue of the discrete Laplacian operator $\nabla^T\nabla$ in L^2 sense. Then with* (6.8) *and* (6.9)*,*

$$(u^k, \Phi^k) \longrightarrow (u^*, \Phi^*)$$

where (u^*, Φ^*) is a saddle point of L in (6.7)*. Define*

$$R^k = (1/\mu)\|u^{k+1} - u^k\|^2 + (1/\tau)\|\Phi^{k+1} - \Phi^k\|^2 - 2\left((\Phi^{k+1} - \Phi^k)^T\nabla^T(u^{k+1} - u^k)\right).$$

Then $R^k \geq 0$ and $R^k = 0$ if and only if (u^k, Φ^k) is a saddle point of (6.7)*. R^k monotonically*

Algorithm 9: $W_{1,p}$ Distance Calculation

Data: discrete density ρ^0, ρ^1; initial u^0 and Φ^0; p; step size μ, τ; and the graph $G = (V, E)$ with orientation encoded in E

for $k = 0, 1, 2, \cdots$, *until converge* **do**
  if $p \neq 1$ **then**
    for $i \in V$ **do**
      $u_i^{k+1} = shrink_p(u_i^k + \mu(\nabla\Phi^k)_i, \mu)$;
    end
  else
    for $(i, j) \in E$ **do**
      $u_{(i,j)}^{k+1} = shrink_1(u_{(i,j)}^k + \mu(\nabla\Phi^k)_{(i,j)}, \mu)$;
    end
  end
  $\Phi^{k+1} = \Phi^k + \tau\left(div(2u^{k+1} - u^k) + \rho^1 - \rho^0\right)$;
end
return u^{k+1}, Φ^{k+1};

Algorithm 10: W_2 Gradient Flow of $W_{1,p}$ distance

Data: the discrete density set S^α; the initial guess set of the saddle point for each $\rho^\beta \in S^\alpha$ $\{(u^{\beta,0}, \Phi^{\beta,0})\}_{\beta=1}^{|S^\alpha|}$; and initial guess of the centroid c^0

for $k = 0, 1, 2, \cdots$, *until converge* **do**
  For each $\rho^\beta \in S^\alpha$, use Algorithm 9 to get $u^{\beta,k+1}, \Phi^{\beta,k+1}$;
  Use (6.14) to get c^{k+1};
end
return c^{k+1}

converges to 0.

And Algorithm 10 is convergent because of the following theorem [53]

Theorem 6.3.3. *Given a simple finite graph $G = (V, E)$ and a functional ρ. The functional $\mathcal{F}(\rho)$ is a Lyapunov function of* (6.13)*. If $\rho(t)$ is a solution of* (6.13) *with initial measure $\rho^0 \in \mathcal{P}(G)$, then*

$$\frac{d}{dt}\mathcal{F}(\rho(t)) \leq 0.$$

At last, we can show the convergence of Algorithm 8.

Theorem 6.3.4. *Algorithm 8 is convergent if the stopping condition is set to be that S does not change in two consecutive iterations.*

Proof. We denote

$$\mathcal{F}(S,c)=\min_{S,c}\sum_{\alpha=1}^{K}\sum_{\rho\in S^{\alpha}}d(\rho,c^{\alpha})$$

where $S=(S^1.\cdots,S^K)$ and $c=(c^1,\cdots,c^K)$. At an arbitrary step, we denote (S_1,c_1) is the result in the previous step and (S_2,c_2) is the result for the current iteration, then by the description of E step, providing that $S_1\neq S_2$, we have $\mathcal{F}(S_2,c_1)\leq\mathcal{F}(S_1,c_1)$. And in M step, if c_1 is a minimizer of $\mathcal{F}(S_2,c)$, we have $c_2=c_1$ and in the next step, since c does not change, S stays at S_2 and the algorithm converges. If $c_2\neq c_1$, by the property of gradient flow, $\mathcal{F}(S_2,c_2)<\mathcal{F}(S_2,c_1)$. Thus, if S changes, $\mathcal{F}(S,c)$ decreases. On the other hand, since the data set if finite, S has a finite number of choices, which leads to the fact that Algorithm 8 converges. □

6.4 Experiments

In this section, we give several examples (Section 6.4.1) to show the calculation and results of $W_{1,p}$ distance with different p and different graph structures. Furthermore we conduct various experiments to illustrate on how the $W_{1,p}$ gradient flow induced by W_2 metric works (Section 6.4.2, 6.4.3 and 6.4.4). At last, examples of Algorithm 8 are used to summarize the performance and results of our approach (Section 6.4.5 and 6.4.6) using $W_{1,2}$ as the distance function. In all the examples, we choose parameters to be $\mu=0.01$, $\tau=10$, initial u and Φ to be zeros vectors.

6.4.1 Calculation of $W_{1,p}$ Distance

We consider a 20×20 grids as our graph and give an orientation, which is shown in Figure 6.1(a). Using different p, we calculate the $W_{1,p}$ distance between two discrete distributions defined on the grid with ρ^0 shown in Figure 6.1(b) a uniform distribution on the red nodes and ρ^1 also in Figure 6.1(b) distributed uniformly on black nodes on the grid. The $W_{1,p}$ distance with different p is given in Table 6.1 and the flux vector u for $p=1,2,3,\infty$

Table 6.1: The $W_{1,p}$ distance with different p given orientation shown in Figure 6.1(a).

p	1	2	3	4	∞
$W_{1,p}$	14.666810	10.606654	9.598984	9.184174	8.451585

Table 6.2: The $W_{1,p}$ distance with different p on the graph in Figure 6.2 with ρ^0, ρ^1 showed in Figure 6.3.

p	1	2	3	4	∞
$W_{1,p}$	13.000826	9.040019	8.018642	7.647420	7.125440

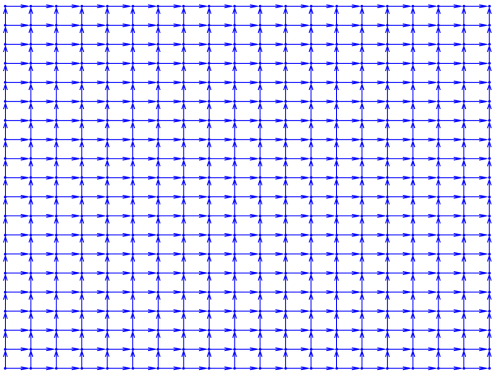
is displayed in Figure 6.1. The orange arrows depict the calculated flux function u with arrows give the direction and the length of the segments describes the intensity.

$W_{1,p}$ also measures the distance on a graph equipped with an orientation. In this section, we use a randomly generated 100-nodes graph, which is given in Figure 6.2. The arrow on the edges are the orientation assigned to carry out the algorithm (Algorithm 9). With different p, we calculate the distance of two distributions uniformly supported on red and black nodes respectively. The arrows and length of orange segments imply the direction and intensity of the flux function u on each edge. Table 6.2 and Figure 6.3 exhibit the results.

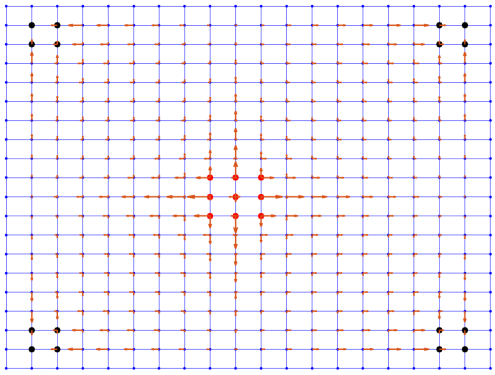
6.4.2 n-point 1-D lattice with one target density

In this subsection, we conduct an experiment on 1-D lattice with 50 points. Denoting the lattice to be $G = (V, E)$ where $V = \{1, 2, \cdots, 50\}$, we assign an orientation on the edge $(i, i+1)$ to be from i to $i+1$ and thus the gradient operator is

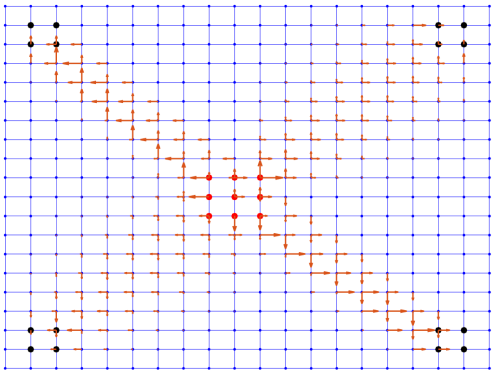
$$\nabla = \begin{pmatrix} -1 & 1 & & & \\ & -1 & 1 & & \\ & & \ddots & \ddots & \\ & & & -1 & 1 \end{pmatrix}. \tag{6.17}$$



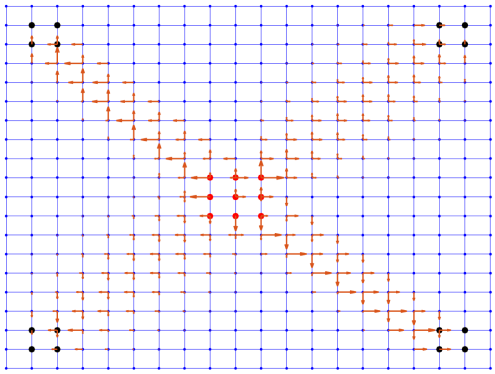
(a) orientation on a 20×20 grid



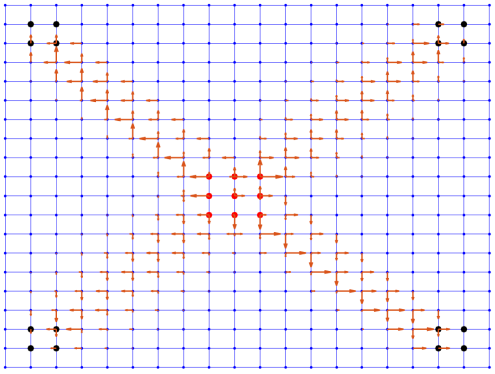
(b) u for $W_{1,1}$ with ρ^0 a uniform distribution on red nodes and ρ^1 a uniform distribution on black nodes. The distance is 14.67.



(c) u for $W_{1,2}$ with ρ^0 a uniform distribution on red nodes and ρ^1 a uniform distribution on black nodes. The distance is 10.61.



(d) u for $W_{1,3}$ with ρ^0 a uniform distribution on red nodes and ρ^1 a uniform distribution on black nodes. The distance is 9.60.



(e) u for $W_{1,\infty}$ with ρ^0 a uniform distribution on red nodes and ρ^1 a uniform distribution on black nodes. The distance is 8.45.

Figure 6.1: The orientation on a 20×20 grid and results of the calculation of $W_{1,p}$ with different p. We break the iteration if $R^k < 1e-8$.

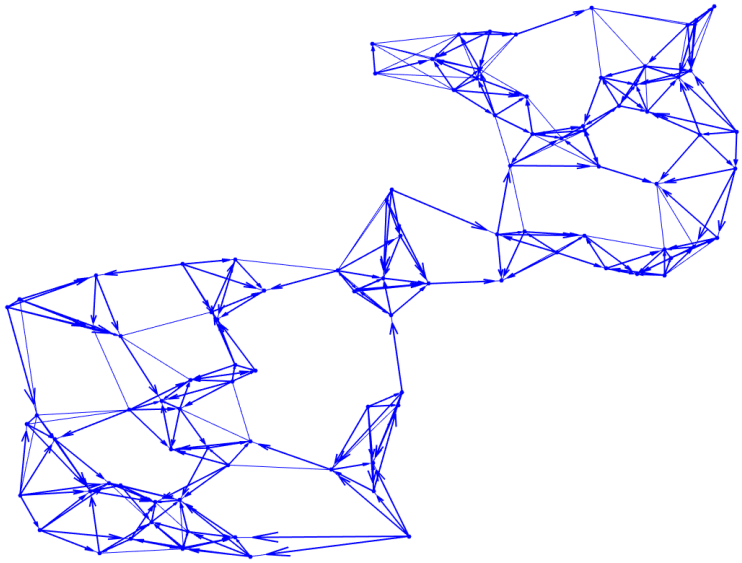


Figure 6.2: Orientation of the graph structure

with all the edge weight to be $d_{(i,j)} = 1$. Now we choose the target density ρ^1 to be

$$\rho_i^1 = \frac{1}{K}\left(\sin\left(\frac{2\pi(i-1)}{49}\right) + 1.1\right)$$

where

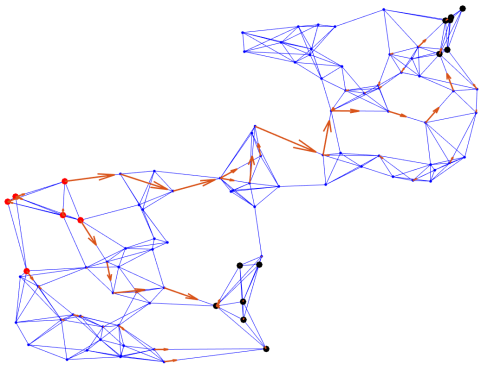
$$K = \sum_{i=1}^{50}\left(\sin\left(\frac{2\pi(i-1)}{49}\right) + 1.1\right)$$

and the objective function can be expressed as

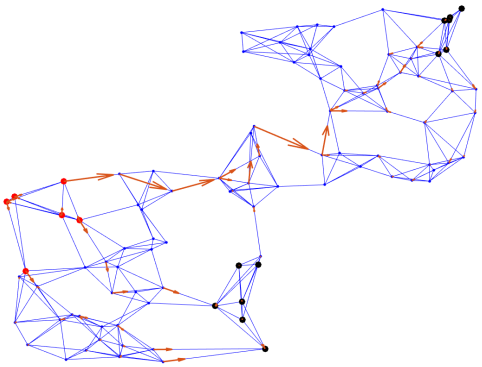
$$\mathcal{F}(\rho) = W_{1,2}(\rho, \rho^1)$$

for which it is obvious that the unique global minimizer if $\rho = \rho^1$. Initially we randomly generate a distribution ρ^0, as shown in Figure 6.4(a), and as evolution conducted by (6.14), ρ^0 approaches ρ^1 gradually and Figure 6.4(b) gives the distance change during the evolution.

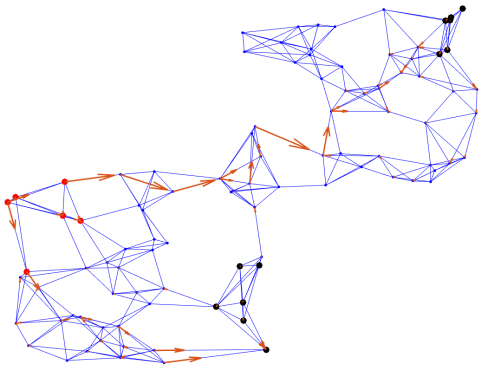
In one full iteration, we need to first calculate the $W_{1,p}$ distance from the current ρ to ρ^1 and then perform an update on ρ, thus the most time consuming step is the computation



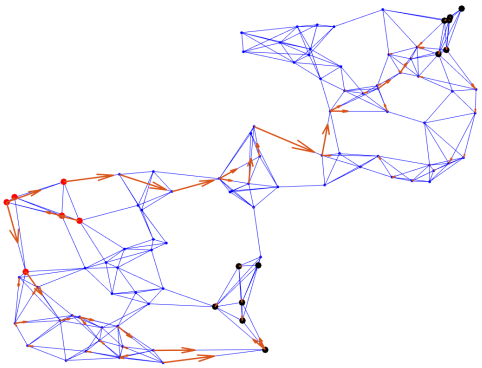
(a) u for $W_{1,1}$ with ρ^0 a uniform distribution on red nodes and ρ^1 a uniform distribution on black nodes. The distance is 13.00.



(b) u for $W_{1,2}$ with ρ^0 a uniform distribution on red nodes and ρ^1 a uniform distribution on black nodes. The distance is 9.04.

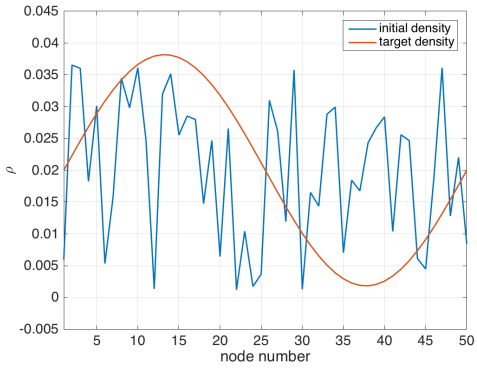


(c) u for $W_{1,3}$ with ρ^0 a uniform distribution on red nodes and ρ^1 a uniform distribution on black nodes. The distance is 8.02.

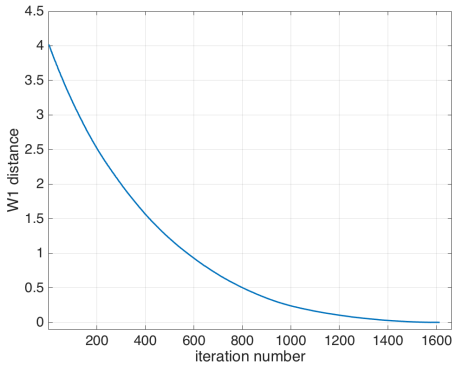


(d) u for $W_{1,\infty}$ with ρ^0 a uniform distribution on red nodes and ρ^1 a uniform distribution on black nodes. The distance is 7.13.

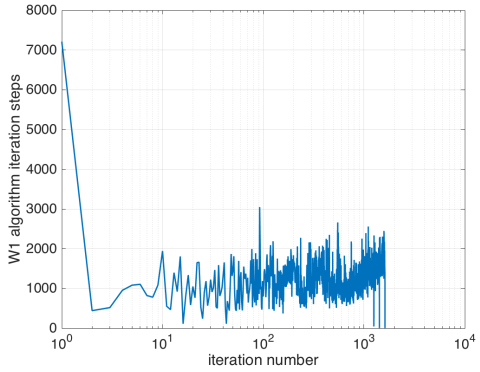
Figure 6.3: The results of the $W_{1,p}$ calculation of $W_{1,p}$ with different p on a general graph given in Figure 6.2. The two distributions are uniformly distributed on the red and black nodes respectively. We break the iteration if $R^k < 1e-8$.



(a) initial and target densities



(b) $W_{1,2}$ distance change



(c) Algorithm 9 iteration steps

Figure 6.4: initial target density functions and result information of example 1. As we can see, the gradient flow drives the initial distribution to the target since the distance goes to 0 and in 6.4(c), the iteration number of Algorithm 9 decreases fast and stay in a low iteration number level during the whole procedure.

step for distances. Fortunately, since in the k-th step, if Δt is relatively small, there is no big change on ρ, which implies that u and Φ in the $W_{1,p}$ calculation will also not change too much because of the continuity of $W_{1,p}$ distance. Therefore, if we use the u^{k-1} and Φ^{k-1} from the $(k-1)$-th step as the initial, the Algorithm 9 converges in a few iterations, which is consistent with the result shown in Figure 6.4(c) that gives the iteration number of Algorithm 9 in each step in Algorithm 10. Furthermore, in Algorithm 8, since all the W_{1p} distance is pre-calculated in Class Assignment step, the gradient flow part does not even need to conduct the most time consuming step, and therefore, Algorithm 8 is efficient.

6.4.3 5-point 1-D lattice with k target density

In this example, a 5 points lattice with $d_{(i,j)} = 1$ and ∇ are defined by (6.17). However, we select a more complicated objective function to deal with,

$$\mathcal{F}(\rho) = \sum_{i=1}^{3} W_{1,2}(\rho, \rho^i)$$

where ρ^i for $i = 1, 2, 3$ are shown in Figure 6.5(a), where

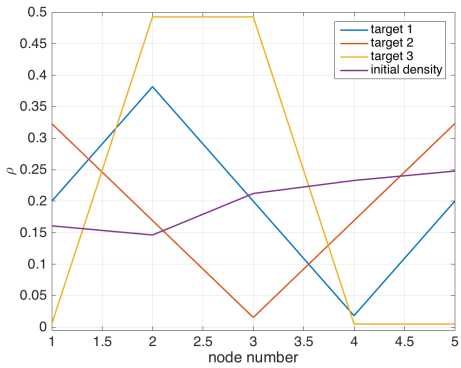
$$\rho_i^1 = \frac{1}{K_1}\left(\sin\left(\frac{2\pi(i-1)}{49}\right) + 1.1\right),$$
$$\rho_i^2 = \frac{1}{K_2}\left(\cos\left(\frac{2\pi(i-1)}{49}\right) + 1.1\right),$$
$$\rho_i^3 = \begin{cases} 100/K_3 & 2 \le i \le 3 \\ 1/K_3 & \text{otherwise} \end{cases}.$$

Again, we randomly generate a distribution on the graph as our initial distribution (in Figure 6.5(a)) and the result given by Algorithm 10 is provided in Figure 6.5(b). Meanwhile, we use the gradient flow induced by Fisher-Rao metric to calculate the problem and we get the same result, which is displayed in Figure 6.5(c). In this example, the W_2 gradient flow shows faster convergence speed compared to the Fisher-Rao gradient flow.

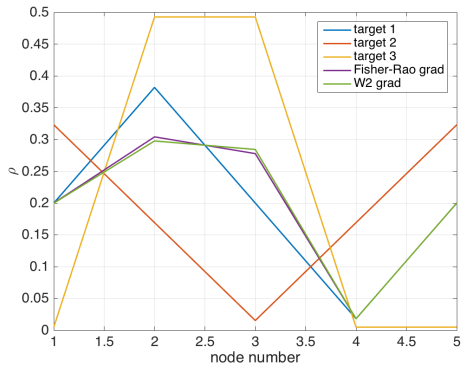
6.4.4 Two components linked by a single edge

To show the different convergence result, we use a graph $G = (V, E)$ displayed in Figure 6.6(a) with vertex label in it. This graph has a feature that it mainly has two separated components and linked by a single edge. Moreover, we assign an arbitrary direction on each edge and form the gradient operator ∇. With the initial distribution and 5 target distributions given in Figure 6.6(b), we assign different sets of edge weights to evolve the W2 gradient flow:

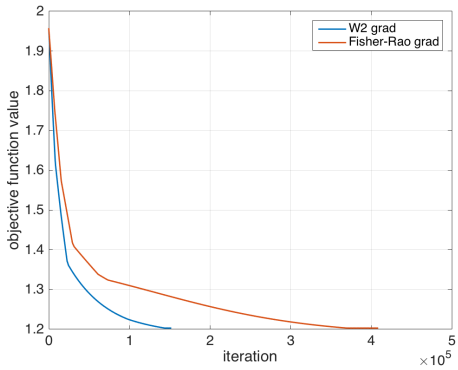
1. $d_{ij} = 1$ for all $(i, j) \in E$;



(a) initial and target densities

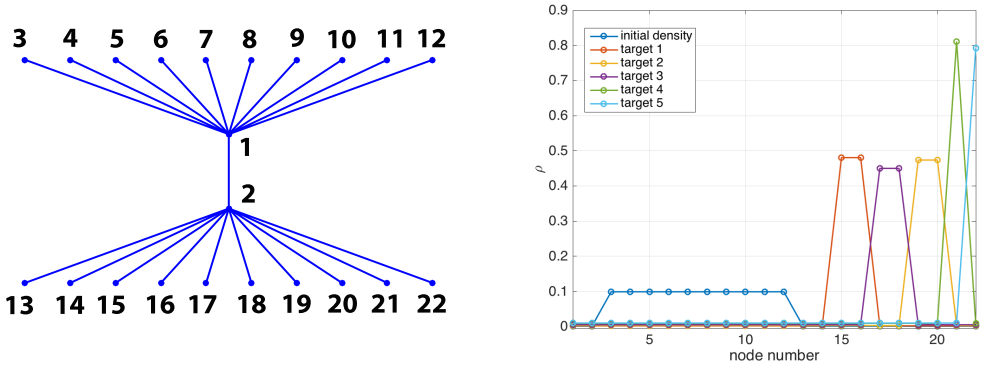


(b) steady state ρ using W_2 gradient flow and Fisher-Rao gradient flow



(c) convergence speed of two gradient flows

Figure 6.5: initial target density functions and steady states with Fisher-Rao and W2 gradient flows of example 2. As we can see, The two methods give us the same results while W_2 gradient flow converges faster



(a) the graph of example 3

(b) initial and target densities

Figure 6.6: The graph and initial target densities for example 3.

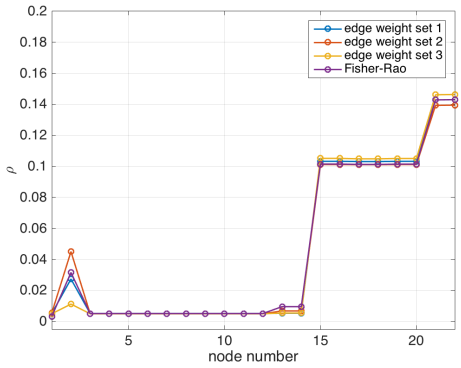
2. $d_{12} = d_{1j} = 20$ for all $j \in N(1)$ and other $d_{ij} = 1$;

3. $d_{12} = d_{2j} = 20$ for all $j \in N(2)$ and other $d_{ij} = 1$.

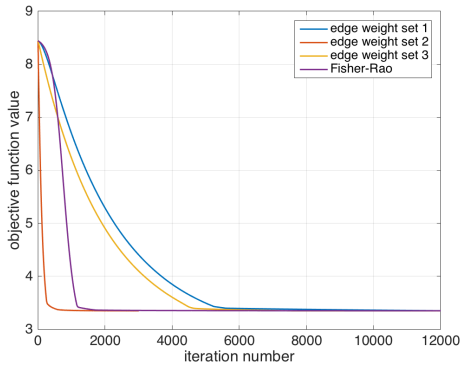
The final steady state results only have slightly difference due to the accuracy of the discretization (6.14) and are given in Figure 6.7(a). The value of the objective function $\mathcal{F}$ as time evolves is shown in Figure 6.7(b). It is shown in this plots that the second set of edge weights gives the fastest convergence result, which converges faster than Fisher-Rao gradient flow. This example indicates the fact that since almost all the mass is transported from the upper part of the graph to the lower part through the edge $(1, 2)$, a higher weight can guarantee a faster transport rate by (6.14). Since initially the mass is uniformly distributed on the upper part of the graph and will simultaneously move to the node 1, a higher weight on $(1, j) \in E$ can ensure the mass rapidly concentrates on the ankle nodes 1 and 2.

Meanwhile, we carry out the gradient flow using the standard projection in L^2 space. As we can see from Figure 6.7(c), the gradient flow with standard Euclidean projection stops converging to the the minimizer after a number of iterations.

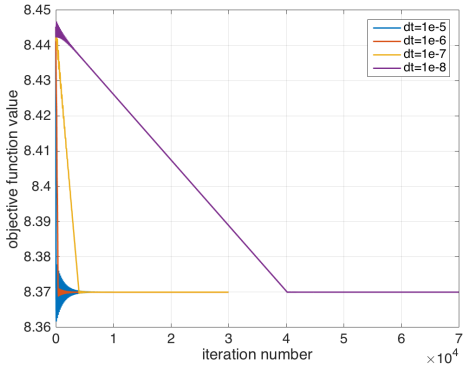
Furthermore, if the initial distribution is on the boundary of $\mathcal{P}(G)$, that is, for some i, $\rho_i^0 = 0$, Fisher-Rao gradient flow will not converge. However, the W_2 gradient flow still works in this case, and if the initial and target are given in Figure 6.8(a), the change of $\rho(t)$ is given in Figure 6.8(b) with the edge weights defined as the second case above. Figure



(a) steady states ρ for different methods

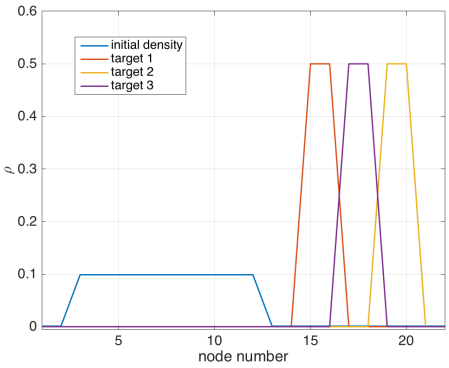


(b) the change of objective function with different methods

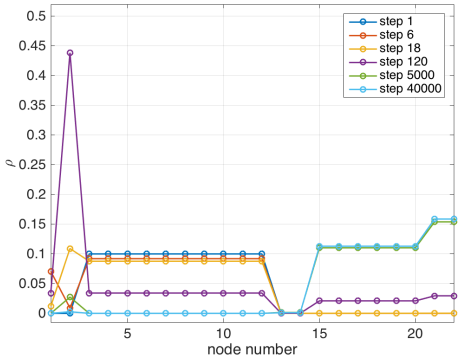


(c) the change of objective function for the gradient flow with standard projection in Euclidean space with different time step size

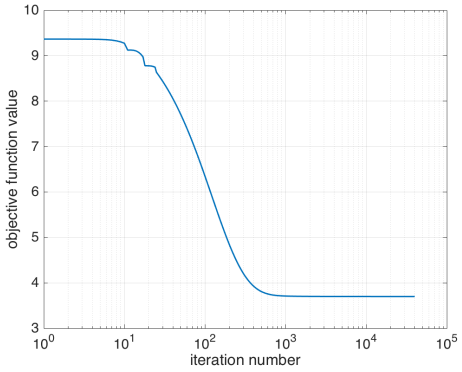
Figure 6.7: Steady states and convergence speed with different methods of example 3. The steady states have slightly difference due to the accuracy. From 6.7(b), we can see that the second set of edge weights gives the fastest convergence result, followed by the forth set, which are both faster than Fisher-Rao gradient flow. And using the standard Euclidean projection is hard to converge.



(a) initial and target densities



(b) ρ in different steps



(c) the change of objective function with different methods

Figure 6.8: initial target density functions and steady states and convergence speed with different methods of example 3 for initial ρ on the boundary case. And 6.8(b) gives several middle steps of the evolution, from which we can see that the evolution always lie on the boundary of $\mathcal{P}(G)$.

6.8(c) gives the trend of $\mathcal{F}$. We observe that the flow $\rho(t)$ will stick on the boundary of the space $\mathcal{P}(G)$ all the time.

Remark 9. *We notice that the accuracy of the convergence is highly related with the discrete time step size Δt in* (6.14)*. This may be caused by the smoothness issue of the L^1 metric implicitly included in the $W_{1,p}$. To better solve* (6.3)*, proximal method may be an approach to handle this issue.*

6.4.5 Clustering problem on a grid

We consider a clustering problem on a 6×6 grid. On it there are 6 discrete distributions that are plotted in Figure 6.9(a) and we want to cluster them into 2 classes. Among them, ρ^1, ρ^2, ρ^3 are delta measures centered around the bottom left corner of the graph and ρ^4, ρ^5, ρ^6 are another 3 delta measures centered around top right corners. We use ρ^1 and ρ^2 as our initial centroids and after 4 iterations, Algorithm 8 separates ρ^1, ρ^2, ρ^3 into one class while ρ^4, ρ^5, ρ^6 into the other class, which is expected. The centroids of the two classes are displayed in Figure 6.9(b). However, if we use Euclidean K-means method, the results varies and one of the results is that $\rho^1, \rho^2, \rho^3, \rho^4, \rho^5$ are clustered in class 1 and ρ^1, ρ^6 are in class 2, with centroids depicted in Figure 6.9(c), 6.9(d). Meanwhile, Algorithm 8 is quite stable for this example and it gives the same results with different initial centroids guesses. We believe that this is because of the good property of Wasserstein-1 distance and the usage of the graph structure when calculating $W_{1,2}$ distance, which are lost if Euclidean distance is involved for clustering.

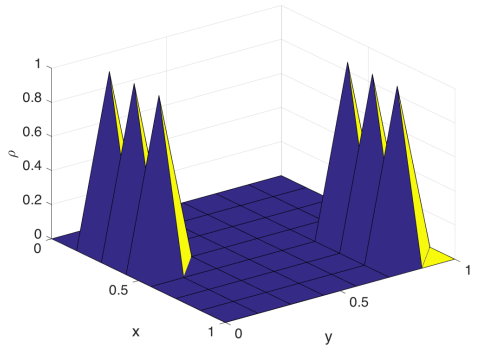
6.4.6 Clustering problem on a general simple graph

In this last example, we randomly generate 100 points P_1 in the box $[0, 0.55] \times [0, 0.55]$ and then randomly generate another 100 points P_2 in $[0.5, 1] \times [0.5, 1]$. Combining the 200 points into the vertex set $V = P_1 \cup P_2 = \{v_i\}_{i=1}^{200}$, we add the edge between $x, y \in V$ if $\|x - y\| \leq 0.1$. Now we have the graph $G = (V, E)$, which is shown in Figure 6.10(a). On the graph, we select 4 points $S_1 = \{x_i\}_{i=1}^{4}$ in P_1 and 4 points $S_2 = \{x_i\}_{i=5}^{8}$ in P_2. Based on $S_1 \cup S_2$, we create our data set as follow:

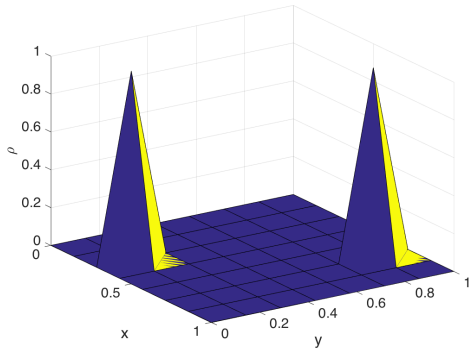
$$\rho_v^i = \frac{1}{K_i} \exp(-100(v - x_i)^2)$$

for $v \in V$ and

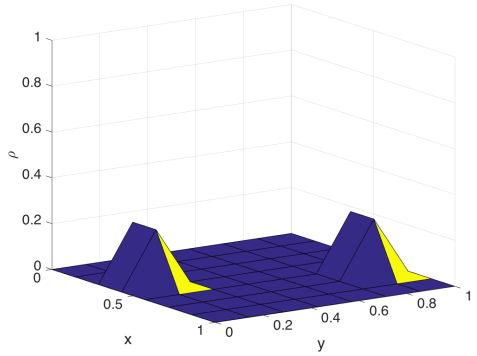
$$K_i = \sum_{v \in V} \exp(-100(v - x_i)^2)$$



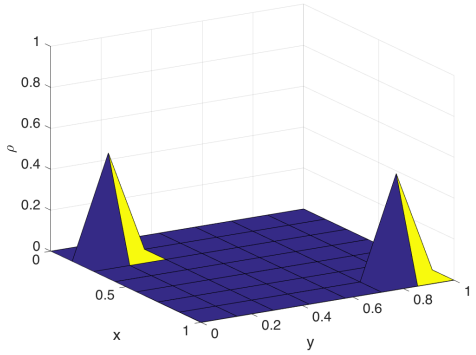
(a) dataset on the 6×6 grid



(b) the centroids derived by proposed method



(c) centroid for class 1 calculated by kmeans in MATLAB



(d) centroid for class 2 calculated by kmeans in MATLAB

Figure 6.9: The data and the centroids. The clustering result is: ρ^1, ρ^2, ρ^3 are in one class while the others are in the other class, which is expected. While kmeans in MATLAB cannot give expected result. If the centroids are shown in Figure 6.9(d) and 6.9(c), ρ^1, ρ^6 are in class 2 and $\rho^1, \rho^2, \rho^3, \rho^4, \rho^5$ are in class 1.

Thus, naturally we have our dataset on the two components of the graph: Figure 6.10(b) shows the densities centered on the lower part of the graph and Figure 6.10(c) gives those concentrated on the upper part, we want to cluster them into 2 classes. Initially we select ρ^2, ρ^7 as the centroids. After the algorithm converges, it gives the centroids as in Figure 6.10(d) and the result is

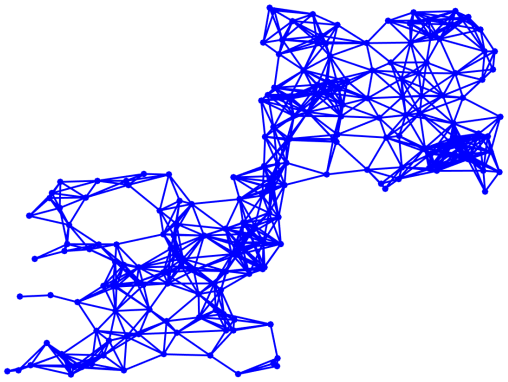
$$C1 = \{\rho^1, \rho^2\}$$
$$C2 = \{\rho^3, \rho^4, \rho^5, \rho^6, \rho^7, \rho^8\}$$

The centroids in the figure concentrate on P_1 and P_2 parts respectively, which is expected. The biased clustering result may be caused by the computational error and the sensitivity of $W_{1,p}$, that is, a small perturbation in the final centroids may cause different clustering results.

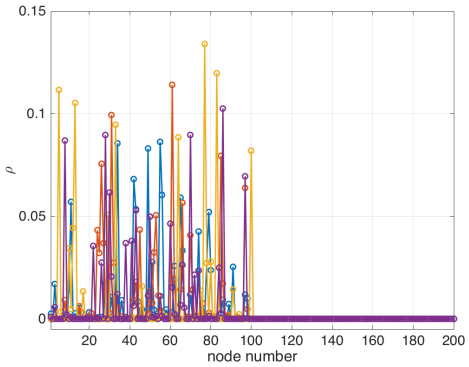
6.5 Conclusion

In this chapter, we define the $W_{1,p}$ distance on graphs with $p \in [1, \infty]$. We model the K-means problem by $W_{1,p}$ distances and apply the Wasserstein-2 gradient descent for the related optimization problems. The algorithm contains two major parts: Calculate $W_{1,p}$ distance and then evaluate the W_2 gradient flow. Each iteration heavily relies on the calculation of $W_{1,p}$ thus we introduce an orientation based definition to make computation easier. Moreover, the results of the previous iteration can be used as the initial condition for current iteration to speedup the convergence of $W_{1,p}$ calculation (Algorithm 9). Meanwhile, this algorithm works when the initial distribution lies on the boundary of $\mathcal{P}(G)$ and can give a reasonable and convergent flow $\rho(t)$.

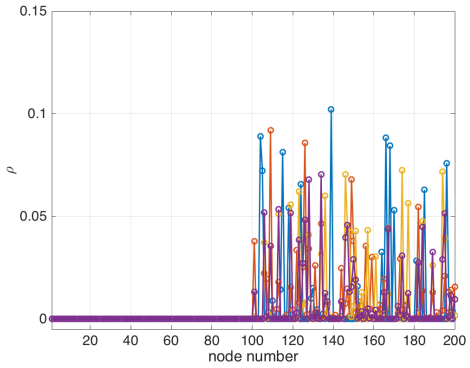
To solve minimization problem (6.3) more efficiently, in the future, we will study the proximal approach instead of the gradient descent method for Wasserstein-1 objective function. This is because the Wasserstein-1 metric is a L_1 type metric, which is not smooth. Be-



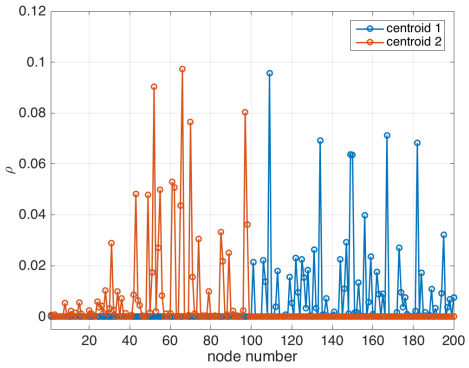
(a) 200 nodes graph on $(0,1) \times (0,1)$



(b) the data set $\{\rho^i\}$ concentrated on the lower part of the graph



(c) the data set $\{\rho^i\}$ concentrated on the upper part of the graph



(d) final centroids

Figure 6.10: The basis information about the clustering problem on a 200 nodes graph with data set size to be 8 and the final centroids with selected initial guess.

sides these computational issues, our approach introduces a new learning modeling method, which naturally incorporates graph structures in datas. Similar approaches has been studied in [114, 115].

REFERENCES

[1] S.-N. Chow, T.-S. Yang, and H.-M. Zhou, "Global optimizations by intermittent diffusion," in *Chaos, CNN, Memristors and Beyond: A Festschrift for Leon Chua With DVD-ROM, composed by Eleonora Bilotta*, World Scientific, 2013, pp. 466–479.

[2] W. Li, J. Lu, H. Zhou, and S.-N. Chow, "Method of evolving junctions: A new approach to optimal control with constraints," *Automatica*, vol. 78, pp. 72–78, 2017.

[3] J. Lu, "Method of evolving junctions: A new approach to path planning and optimal control," PhD thesis, Georgia Institute of Technology, 2014.

[4] M. H. Overmars, *A random approach to motion planning*. Unknown Publisher, 1992, vol. 92.

[5] L. Kavraki and J.-C. Latombe, "Randomized preprocessing of configuration for fast path planning," in *Robotics and Automation, 1994. Proceedings., 1994 IEEE International Conference on*, IEEE, 1994, pp. 2138–2145.

[6] N. M. Amato and Y. Wu, "A randomized roadmap method for path and manipulation planning," in *Robotics and Automation, 1996. Proceedings., 1996 IEEE International Conference on*, IEEE, vol. 1, 1996, pp. 113–120.

[7] P. Svestka, "Robot motion planning using probabilistic roadmaps," *PhD Thesis, Universiteit Utrecht*, 1997.

[8] D. Hsu, J.-C. Latombe, and R. Motwani, "Path planning in expansive configuration spaces," in *Robotics and Automation, 1997. Proceedings., 1997 IEEE International Conference on*, IEEE, vol. 3, 1997, pp. 2719–2726.

[9] J. Barraquand, L. Kavraki, J.-C. Latombe, R. Motwani, T.-Y. Li, and P. Raghavan, "A random sampling scheme for path planning," *The International Journal of Robotics Research*, vol. 16, no. 6, pp. 759–774, 1997.

[10] M. S. Branicky, S. M. LaValle, K. Olson, and L. Yang, "Quasi-randomized path planning," in *Robotics and Automation, 2001. Proceedings 2001 ICRA. IEEE International Conference on*, IEEE, vol. 2, 2001, pp. 1481–1487.

[11] V. Boor, M. H. Overmars, and A. F. Van Der Stappen, "The gaussian sampling strategy for probabilistic roadmap planners," in *Robotics and automation, 1999.*

proceedings. 1999 ieee international conference on, IEEE, vol. 2, 1999, pp. 1018–1023.

[12] S. M. LaValle, "Rapidly-exploring random trees: A new tool for path planning," 1998.

[13] Y. Tian, L. Yan, G.-Y. Park, S.-H. Yang, Y.-S. Kim, S.-R. Lee, and C.-Y. Lee, "Application of rrt-based local path planning algorithm in unknown environment," in *Computational Intelligence in Robotics and Automation, 2007. CIRA 2007. International Symposium on*, IEEE, 2007, pp. 456–460.

[14] K. Yang, S. Keat Gan, and S. Sukkarieh, "A gaussian process-based rrt planner for the exploration of an unknown and cluttered environment with a uav," *Advanced Robotics*, vol. 27, no. 6, pp. 431–443, 2013.

[15] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *The international journal of robotics research*, vol. 30, no. 7, pp. 846–894, 2011.

[16] O. Salzman and D. Halperin, "Asymptotically near-optimal rrt for fast, high-quality motion planning," *IEEE Transactions on Robotics*, vol. 32, no. 3, pp. 473–483, 2016.

[17] I. Noreen, A. Khan, and Z. Habib, "Optimal path planning using rrt* based approaches: A survey and future directions," *Int. J. Adv. Comput. Sci. Appl*, vol. 7, pp. 97–107, 2016.

[18] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," in *Autonomous robot vehicles*, Springer, 1986, pp. 396–404.

[19] J. Sfeir, M. Saad, and H. Saliah-Hassane, "An improved artificial potential field approach to real-time mobile robot path planning in an unknown environment," in *Robotic and Sensors Environments (ROSE), 2011 IEEE International Symposium on*, IEEE, 2011, pp. 208–213.

[20] K. Ahlin, "The secant and traveling artificial potential field approaches to high dimensional robotic path planning," *PhD Thesis, Georgia Institute of Technology*, 2018.

[21] Y.-b. Chen, G.-c. Luo, Y.-s. Mei, J.-q. Yu, and X.-l. Su, "Uav path planning using artificial potential field method updated by optimal control theory," *International Journal of Systems Science*, vol. 47, no. 6, pp. 1407–1420, 2016.

[22] O. Montiel, U. Orozco-Rosas, and R. Sepúlveda, "Path planning for mobile robots using bacterial potential field for avoiding static and dynamic obstacles," *Expert Systems with Applications*, vol. 42, no. 12, pp. 5177–5191, 2015.

[23] V Lumelsky and A. Stepanov, "Dynamic path planning for a mobile automaton with limited information on the environment," *IEEE transactions on Automatic control*, vol. 31, no. 11, pp. 1058–1063, 1986.

[24] V. J. Lumelsky and A. A. Stepanov, "Path-planning strategies for a point mobile automaton moving amidst unknown obstacles of arbitrary shape," *Algorithmica*, vol. 2, no. 1-4, pp. 403–430, 1987.

[25] I. Kamon, E. Rivlin, and E. Rimon, "A new range-sensor based globally convergent navigation algorithm for mobile robots," in *Proceedings of IEEE International Conference on Robotics and Automation*, IEEE, vol. 1, 1996, pp. 429–435.

[26] I. Kamon and E. Rivlin, "Sensory-based motion planning with global proofs," *IEEE transactions on Robotics and Automation*, vol. 13, no. 6, pp. 814–822, 1997.

[27] K. McGuire, G. de Croon, and K. Tuyls, "A comparative study of bug algorithms for robot navigation," *arXiv preprint arXiv:1808.05050*, 2018.

[28] J. Ng and T. Bräunl, "Performance comparison of bug navigation algorithms," *Journal of Intelligent and Robotic Systems*, vol. 50, no. 1, pp. 73–84, 2007.

[29] J. E. Doran and D. Michie, "Experiments with the graph traverser program," *Proceedings of the Royal Society of London. Series A. Mathematical and Physical Sciences*, vol. 294, no. 1437, pp. 235–259, 1966.

[30] A. Stentz, "Optimal and efficient path planning for partially-known environments," in *ICRA*, vol. 94, 1994, pp. 3310–3317.

[31] A. Stentz *et al.*, "The focussed $\hat{\text{d}}^*$ algorithm for real-time replanning," in *IJCAI*, vol. 95, 1995, pp. 1652–1659.

[32] S. Koenig and M. Likhachev, "Fast replanning for navigation in unknown terrain," *IEEE Transactions on Robotics*, vol. 21, no. 3, pp. 354–363, 2005.

[33] L. Podsedkowski, J. Nowakowski, M. Idzikowski, and I. Vizvary, "A new solution for path planning in partially known or unknown environment for nonholonomic mobile robots," *Robotics and Autonomous Systems*, vol. 34, no. 2-3, pp. 145–152, 2001.

[34] M. Walker and C. H. Messom, "A comparison of genetic programming and genetic algorithms for auto-tuning mobile robot motion control," in *Electronic Design, Test*

and Applications, 2002. Proceedings. The First IEEE International Workshop on, IEEE, 2002, pp. 507–509.

[35] L. Lei, H. Wang, and Q. Wu, "Improved genetic algorithms based path planning of mobile robot under dynamic unknown environment," in *Mechatronics and Automation, Proceedings of the 2006 IEEE International Conference on*, IEEE, 2006, pp. 1728–1732.

[36] M. A. Contreras-Cruz, V. Ayala-Ramirez, and U. H. Hernandez-Belmonte, "Mobile robot path planning using artificial bee colony and evolutionary programming," *Applied Soft Computing*, vol. 30, pp. 319–328, 2015.

[37] I. Hassanzadeh and S. M. Sadigh, "Path planning for a mobile robot using fuzzy logic controller tuned by ga," in *Mechatronics and its Applications, 2009. ISMA'09. 6th International Symposium on*, IEEE, 2009, pp. 1–5.

[38] M. Wang *et al.*, "Fuzzy logic based robot path planning in unknown environment," in *Machine Learning and Cybernetics, 2005. Proceedings of 2005 International Conference on*, IEEE, vol. 2, 2005, pp. 813–818.

[39] C. Luo and S. X. Yang, "A bioinspired neural network for real-time concurrent map building and complete coverage robot navigation in unknown environments," *IEEE Transactions on Neural Networks*, vol. 19, no. 7, pp. 1279–1298, 2008.

[40] T. Ersson and X. Hu, "Path planning and navigation of mobile robots in unknown environments.," in *IROS*, 2001, pp. 858–864.

[41] L. Janson, E. Schmerling, A. Clark, and M. Pavone, "Fast marching tree: A fast marching sampling-based method for optimal motion planning in many dimensions," *The International journal of robotics research*, vol. 34, no. 7, pp. 883–921, 2015.

[42] D. Dolgov, S. Thrun, M. Montemerlo, and J. Diebel, "Path planning for autonomous vehicles in unknown semi-structured environments," *The International Journal of Robotics Research*, vol. 29, no. 5, pp. 485–501, 2010.

[43] J. Van Den Berg, D. Ferguson, and J. Kuffner, "Anytime path planning and replanning in dynamic environments," in *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on*, IEEE, 2006, pp. 2366–2371.

[44] G. Wagner and H. Choset, "Subdimensional expansion for multirobot path planning," *Artificial Intelligence*, vol. 219, pp. 1–24, 2015.

[45] R. Cui, Y. Li, and W. Yan, "Mutual information-based multi-auv path planning for scalar field sampling using multidimensional rrt," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 46, no. 7, pp. 993–1004, 2016.

[46] P. Das, H. S. Behera, and B. K. Panigrahi, "A hybridization of an improved particle swarm optimization and gravitational search algorithm for multi-robot path planning," *Swarm and Evolutionary Computation*, vol. 28, pp. 14–28, 2016.

[47] S. Bandyopadhyay, S.-J. Chung, and F. Y. Hadaegh, "Probabilistic swarm guidance using optimal transport," in *2014 IEEE Conference on Control Applications (CCA)*, IEEE, 2014, pp. 498–505.

[48] V. Krishnan and S. Martínez, "Distributed optimal transport for the deployment of swarms," in *2018 IEEE Conference on Decision and Control (CDC)*, IEEE, 2018, pp. 4583–4588.

[49] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numerische mathematik*, vol. 1, no. 1, pp. 269–271, 1959.

[50] K. Zuse, *Der Plankalkül*, 63. Gesellschaft für Mathematik und Datenverarbeitung, 1972.

[51] E. F. Moore, "The shortest path through a maze," in *Proc. Int. Symp. Switching Theory, 1959*, 1959, pp. 285–292.

[52] S.-N. Chow, W. Huang, Y. Li, and H. Zhou, "Fokker–planck equations for a free energy functional or markov process on a graph," *Archive for Rational Mechanics and Analysis*, vol. 203, no. 3, pp. 969–1008, 2012.

[53] W. Li, "A study of stochastic differential equations and fokker-planck equations with applications," PhD thesis, Georgia Institute of Technology, 2016.

[54] E. D. Sontag, *Mathematical control theory: deterministic finite dimensional systems*. Springer Science & Business Media, 2013, vol. 6.

[55] N. Jacobson, *Lie algebras*, 10. Courier Corporation, 1979.

[56] N. E. Leonard, D. A. Paley, R. E. Davis, D. M. Fratantoni, F. Lekien, and F. Zhang, "Coordinated control of an underwater glider fleet in an adaptive ocean sampling field experiment in Monterey Bay," *Journal of Field Robotics*, vol. 27, no. 6, pp. 718–740, 2010.

[57] R. N. Smith, Y. Chao, P. P. Li, D. A. Caron, B. H. Jones, and G. S. Sukhatme, "Planning and implementing trajectories for autonomous underwater vehicles to track evolving ocean processes based on predictions from a Regional Ocean Model,"

The International Journal of Robotics Research, vol. 29, no. 12, pp. 1475–1497, 2010.

[58] P. Ozog, N. Carlevaris-Bianco, A. Y. Kim, and R. M. Eustice, "Long-term mapping techniques for ship hull inspection and surveillance using an autonomous underwater vehicle," *J. Field Robotics*, vol. 33, pp. 265–289, 2016.

[59] B. Rhoads, I. Mezic, and A. C. Poje, "Minimum time heading control of underpowered vehicles in time-varying ocean currents," *Ocean Engineering*, vol. 66, no. 1, pp. 12–31, 2012.

[60] A. A. Pereira, J. Binney, G. A. Hollinger, and G. S. Sukhatme, "Risk-aware path planning for autonomous underwater vehicles using predictive ocean models," *Journal of Field Robotics*, vol. 30, no. 5, pp. 741–762, 2013.

[61] S. M. Lavalle, "Rapidly-Exploring Random Trees: A new tool for path planning," Department of Computer Science, Iowa State University, Tech. Rep., 1998.

[62] J. Kuffner and S. LaValle, "RRT-connect: An efficient approach to single-query path planning," in *IEEE International Conference on Robotics and Automation*, 2000.

[63] J. A. Sethian, *Level Set Methods and Fast Marching Methods: Evolving Interfaces in Geometry, Fluid Mechanics, Computer Vison and Material Science*. Cambridge University Press, 1999.

[64] S. V. T. Lolla, "Path planning and adaptive sampling in the coastal ocean," PhD thesis, Massachusetts Institute of Technology, 2016.

[65] M. Soulignac, "Feasible and optimal path planning in strong current fields," *IEEE Transactions on Robotics*, vol. 27, no. 1, pp. 89–98, 2011.

[66] D. A. Paley, "Cooperative control of collective motion for ocean sampling with autonomous vehicles," Ph. D. thesis, Princeton University, 2007.

[67] F. Zhang, D. M. Fratantoni, D. A. Paley, J. M. Lund, and N. E. Leonard, "Control of coordinated patterns for ocean sampling," *International Journal of Control*, vol. 80, no. 7, pp. 1186–1199, 2007.

[68] F. Zhang, G. Marani, R. N. Smith, and H. T. Choi, "Future trends in marine robotics," *IEEE Robotics and Automation Magazine*, vol. 22, no. 1, 14–21, pp. 122, 2015.

[69] E. Fiorelli, N. E. Leonard, P. Bhatta, D. A. Paley, R. Bachmayer, and D. M. Fratantoni, "Multi-AUV control and adaptive sampling in Monterey Bay," *IEEE Journal of Oceanic Engineering*, vol. 31, no. 4, pp. 935–948, 2006.

[70] S. Liu, J. Sun, J. Yu, A. Zhang, and F. Zhang, "Distributed traversability analysis of flow field under communication constraints," *IEEE Journal of Oceanic Engineering*, 2018.

[71] D. B. Gennery, "Traversability analysis and path planning for a planetary rover," *Autonomous Robots*, vol. 6, no. 2, pp. 131–146, 1999.

[72] A. Howard and H. Seraji, "Vision-based terrain characterization and traversability assessment," *Journal of Robotic Systems*, vol. 18, pp. 577 –587, Oct. 2001.

[73] A. E. Bryson and Y. Ho, *Applied Optimal Control: Optimization, Estimation and Control*. CRC Press, 1975.

[74] D. Subramani and P. Lermusiaux, "Energy–optimal path planning by stochastic dynamically orthogonal level–set optimization," *Ocean Modelling*, vol. 100, Feb. 2016.

[75] S. Lolla, "Path planning in time dependent flows using level set method," M.S. thesis, Massachusetts Institute of Technology, 2012.

[76] I. M. Mitchell, "A toolbox for Level Set Method," University of British Columbia, Department of Computer Science, Tech. Rep., 2007.

[77] D. Adalsteinsson and J. A. Sethian, "A fast level set method for propagating interfaces," *Journal of Computational Physics*, vol. 118, no. 2, pp. 269 –277, 1995.

[78] P. J. Martin, "Description of the navy coastal ocean model version 1.0," Naval Research Lab, Tech. Rep. NRL/FR/7322–00-9962, 2000.

[79] J. MacQueen *et al.*, "Some methods for classification and analysis of multivariate observations," in *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, Oakland, CA, USA, vol. 1, 1967, pp. 281–297.

[80] J. A. Hartigan and M. A. Wong, "Algorithm as 136: A k-means clustering algorithm," *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, vol. 28, no. 1, pp. 100–108, 1979.

[81] S. Lloyd, "Least squares quantization in pcm," *IEEE transactions on information theory*, vol. 28, no. 2, pp. 129–137, 1982.

[82] M. Garey, D Johnson, and H. Witsenhausen, "The complexity of the generalized lloyd-max problem (corresp.)," *IEEE Transactions on Information Theory*, vol. 28, no. 2, pp. 255–256, 1982.

[83] D. Aloise, A. Deshpande, P. Hansen, and P. Popat, "Np-hardness of euclidean sum-of-squares clustering," *Machine learning*, vol. 75, no. 2, pp. 245–248, 2009.

[84] D. Arthur and S. Vassilvitskii, "K-means++: The advantages of careful seeding," in *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, Society for Industrial and Applied Mathematics, 2007, pp. 1027–1035.

[85] C. Elkan, "Using the triangle inequality to accelerate k-means," in *Proceedings of the 20th International Conference on Machine Learning (ICML-03)*, 2003, pp. 147–153.

[86] Y. Ding, Y. Zhao, X. Shen, M. Musuvathi, and T. Mytkowicz, "Yinyang k-means: A drop-in replacement of the classic k-means with consistent speedup," in *International Conference on Machine Learning*, 2015, pp. 579–587.

[87] P. Yin, M. Pham, A. Oberman, and S. Osher, "Stochastic backward euler: An implicit gradient descent algorithm for k-means clustering," *Journal of Scientific Computing*, vol. 77, no. 2, pp. 1133–1146, 2018.

[88] S. Gupta, R. Kumar, K. Lu, B. Moseley, and S. Vassilvitskii, "Local search methods for k-means with outliers," *Proceedings of the VLDB Endowment*, vol. 10, no. 7, pp. 757–768, 2017.

[89] J. Wang, J. Wang, Q. Ke, G. Zeng, and S. Li, "Fast approximate k k-means via cluster closures," in *Multimedia data mining and analytics*, Springer, 2015, pp. 373–395.

[90] C. Villani, *Topics in optimal transportation*, 58. American Mathematical Soc., 2003.

[91] N. Ho, X. Nguyen, M. Yurochkin, H. H. Bui, V. Huynh, and D. Phung, "Multilevel clustering via wasserstein means," *arXiv preprint arXiv:1706.03883*, 2017.

[92] M. Staib and S. Jegelka, "Wasserstein k-means++ for cloud regime histogram clustering," *Climate Informatics*, 2017.

[93] L. Mi, W. Zhang, X. Gu, and Y. Wang, "Variational wasserstein clustering," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 322–337.

[94] S. Kolouri, Y. Zou, and G. K. Rohde, "Sliced wasserstein kernels for probability distributions," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 5258–5267.

[95] A. Irpino, R. Verde, and Y. Lechevallier, "Dynamic clustering of histograms using wasserstein metric," in *COMPSTAT*, 2006, pp. 869–876.

[96] A. Irpino and R. Verde, "Dynamic clustering of interval data using a wasserstein-based distance," *Pattern Recognition Letters*, vol. 29, no. 11, pp. 1648–1658, 2008.

[97] N. Bonneel, G. Peyré, and M. Cuturi, "Wasserstein barycentric coordinates: Histogram regression using optimal transport.," *ACM Trans. Graph.*, vol. 35, no. 4, pp. 71–1, 2016.

[98] J. Solomon, F. De Goes, G. Peyré, M. Cuturi, A. Butscher, A. Nguyen, T. Du, and L. Guibas, "Convolutional wasserstein distances: Efficient optimal transportation on geometric domains," *ACM Transactions on Graphics (TOG)*, vol. 34, no. 4, p. 66, 2015.

[99] J. Solomon, R. Rustamov, L. Guibas, and A. Butscher, "Earth mover's distances on discrete surfaces," *ACM Transactions on Graphics (TOG)*, vol. 33, no. 4, p. 67, 2014.

[100] J. Gu, B. Hua, and S. Liu, "Spectral distances on graphs," *Discrete Applied Mathematics*, vol. 190, pp. 56–74, 2015.

[101] S. P. Singh, A. Hug, A. Dieuleveut, and M. Jaggi, "Context mover's distance & barycenters: Optimal transport of contexts for building representations," 2018.

[102] J. Chen, H. Matzinger, H. Zhai, and M. Zhou, "Centroid estimation based on symmetric kl divergence for multinomial text classification problem," in *2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA)*, IEEE, 2018, pp. 1174–1177.

[103] N. Friedman, D. Geiger, and M. Goldszmidt, "Bayesian network classifiers," *Machine learning*, vol. 29, no. 2-3, pp. 131–163, 1997.

[104] P. Langley, W. Iba, K. Thompson, *et al.*, "An analysis of bayesian classifiers," in *Aaai*, vol. 90, 1992, pp. 223–228.

[105] W. Li, E. K. Ryu, S. Osher, W. Yin, and W. Gangbo, "A parallel method for earth mover's distance," *Journal of Scientific Computing*, vol. 75, no. 1, pp. 182–197, 2018.

[106] S.-N. Chow, W. Li, and H. Zhou, "Entropy dissipation of Fokker-Planck equations on graphs," *Discrete & Continuous Dynamical Systems, series A*, 2018. arXiv: 1701.04841.

[107] S.-N. Chow, L. Dieci, W. Li, and H. Zhou, "Entropy dissipation semi-discretization schemes for fokker–planck equations," *Journal of Dynamics and Differential Equations*, pp. 1–28, 2018.

[108] J.-D. Benamou and G. Carlier, "Augmented lagrangian methods for transport optimization, mean field games and degenerate elliptic equations," *Journal of Optimization Theory and Applications*, vol. 167, no. 1, pp. 1–26, 2015.

[109] J.-D. Benamou and Y. Brenier, "A computational fluid mechanics solution to the monge-kantorovich mass transfer problem," *Numerische Mathematik*, vol. 84, no. 3, pp. 375–393, 2000.

[110] M. Beckmann, "A continuous model of transportation," *Econometrica: Journal of the Econometric Society*, pp. 643–660, 1952.

[111] A. Chambolle and T. Pock, "A first-order primal-dual algorithm for convex problems with applications to imaging," *Journal of mathematical imaging and vision*, vol. 40, no. 1, pp. 120–145, 2011.

[112] T. Pock and A. Chambolle, "Diagonal preconditioning for first order primal-dual algorithms in convex optimization," in *Computer Vision (ICCV), 2011 IEEE International Conference on*, IEEE, 2011, pp. 1762–1769.

[113] P. Milgrom and I. Segal, "Envelope theorems for arbitrary choice sets," *Econometrica*, vol. 70, no. 2, pp. 583–601, 2002.

[114] Y. Dukler, W. Li, A. Lin, and G. Montufar, *Wasserstein of wasserstein loss for learning generative models*, 2019.

[115] A. Lin, W. Li, S. Osher, and G. Montufar, *Wasserstein proximal of GANs*, 2019.